(12)

Western Kentucky University

AD A056270

LEVEL II

AN APPLICATION OF OPTIMAL CONTROL THEORY TO AN ANTI-TANK WEAPON SYSTEM

DDC FILE COPY

DDC
RECEIVED
JUL 14 1978
A

Principle Researchers : Dr. Randy J. York
Dr. Daniel C. St.Clair
Department of Mathematics
and Computer Science

Research Assistant : Peter Sisler

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

78 07 14 052

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Application of Optimal Control Theory to an Anti-Tank Weapon System		5. TYPE OF REPORT & PERIOD COVERED Final Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Randy J. York, Daniel C. St. Clair		8. CONTRACT OR GRANT NUMBER(s) DAAK40-77-C-0052
9. PERFORMING ORGANIZATION NAME AND ADDRESS Western Kentucky University, Bowling Green, Kentucky 42101		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander; U.S. Army Missile Research and Development Command, ATTN: DRDMI-TGN, Redstone Arsenal, AL 35809		12. REPORT DATE 1 June 1978
		13. NUMBER OF PAGES 165
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optimal control law formulation
Optimal control law implementation
Terminal guidance
Anti-Tank weapon system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem of implementing a three-state optimal control law in an anti-tank weapon system was examined. The weapon system was represented by a detailed six degree of freedom simulation.

The derivation of the control law is included along with a four-state simulation that was used to model the 6DOF simulation. The procedure used to implement the control law is given in Chapter III, and the method of obtaining optimal control parameters appears in Appendix C. The question of sensitivity is also examined using both simulations.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(6) (9) (10) (11) (12) 172p. (15) 384708

78 07 14 052

June 1, 1978

FINAL REPORT

for

Western Kentucky University Contract # DAAK 40-77-C-0052
(Redstone Arsenal Contract # ~~PAN~~ TB-33)
RA

AN APPLICATION OF OPTIMAL CONTROL THEORY TO AN ANTI-TANK WEAPON SYSTEM

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

to

Harold Pastrick, Contract Supervisor
Guidance and Control Directorate
Research and Development Laboratory
U. S. Army Missile Command
Redstone Arsenal, Alabama 35809

from

Dr. Randy J. York and Dr. Daniel C. St. Clair
Department of Mathematics and Computer Science
Western Kentucky University
Bowling Green, Kentucky 42101

Peter Sisler, research assistant

ABSTRACT

The problem of implementing a three-state optimal control law in a anti-tank weapon system was examined. The weapon system was represented by a detailed six degree of freedom simulation.

The derivation of the control law is included along with a four-state simulation that was used to model the 6DOF simulation. The procedure used to implement the control law is given in Chapter III, and the method of obtaining optimal control parameters appears in Appendix C. The question of sensitivity is also examined using both simulations. Final conclusions and recommendations appear in Chapter V.

ACKNOWLEDGEMENTS

Dr. Harold Pastrick, contract supervisor, of the U.S. Army Missile Command, has been most helpful with his suggestions and demands. We would also like to acknowledge the assistance of Dr. Thomas Madron and his Research and Computer Services group at Western who have patiently answered our many questions and requests. This contract, with its security requirement, has posed special problems, and we would like to thank Dr. Glenn Crumb and his assistants at the Office of Grants and Contracts for their aid.

This report could not have been prepared without the help of Debra Wheeler, who not only was most careful in the typing of this manuscript, but who was able to spot and correct oversights on our part.

Someone has to be last, but we hope that no affront is taken. We would like to especially thank our wives for their patience and understanding they showed during this contractual period and the preparation of this final report.

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter I

INTRODUCTION

1.1 Background

Recent intelligence suggests that the impenetrable nature of heavy armor may be susceptible to missile attacks at a relatively high angle of impact, with respect to the horizon. In many modes of direct encounter, the target may not be reachable with a body pitch attitude angle of the proper magnitude. There are several possible reasons for this condition, including lack of energy (fuel), lack of time to maneuver into the more desirable attitude, or lack of control information by appropriate sensors to command the response. This condition has been recognized for some time at the Missile Research and Development Command, and consequently there have been attempts to modify trajectory shapes by a variety of predetermined control laws. However, there has been a certain lack of robustness in the solutions obtained over the entire range of conditions anticipated. This situation motivated a search for optimal solutions to the guidance problem and a study of tradeoffs among the suboptimal candidates that were deemed feasible.

Terminal guidance schemes for tactical missiles may be based on a classical approach, such as a proportional navigation and guidance law [3,4], or on a modern control theoretic approach [1,5,6]. In the latter, a control law is derived in terms of time-varying feedback gains when formulated as a linear quadratic control problem. A sub-optimal terminal guidance system for re-entry vehicles, derived using the modern approach, was the basis for the initial work of this problem.

Kim and Grider [2] studied a suboptimal terminal guidance system for a re-entry vehicle by placing a constraint on the body attitude angle at impact. Their problem was oriented to a long-range, high-altitude mission. Their scenario was formulated as a linear quadratic control problem with certain key assumptions. The angle of attack of the re-entry vehicle was assumed to be small and thus was neglected. Furthermore, the autopilot response was assumed to be instantaneous, i.e., with no lag time attributed to the transfer of input commands to output reaction.

These conditions have been studied in an extension of their earlier work where a formulation is given for a system that has finite time delay. In fact, the increase and decrease in time delay has interesting ramifications on the solution. The angle-of-attack assumption is investigated, and, although not solved analytically in closed form, the system is derived [9].

There is more than just a passing academic interest in this problem. As suggested previously, the antiarmor role of several Army weapon systems very well may be enhanced by this technique. The reduction to a practical implementation or mechanization is the aim of this contract.

1.2 Implementation of the Control Law

In order to investigate the problems arising from the practical implementation of such an optimal control law on a real system, a detailed six degree of freedom (6DOF) digital simulation of such a system was needed. Such simulations are so complex that to implement the control law from just a programming point of view would require that the simulation would have to be in modular form; otherwise, the programming difficulties would obscure the implementation difficulties. Such a simulation of a semiactive laser-guided air-to-ground missile was provided to the authors by Dr. Harold L. Pastrick of the Army Missile Command, the contract supervisor. Unfortunately, none was available that would work on the IBM370 system at the University of Kentucky that would be used - see section 2.1 for a discussion of the considerable difficulties encountered.

The detailed 6DOF simulation would give an accurate representation of the real system and how it would respond with the new controller. Although a simpler four-state simulation would be used as a guide in implementing the control law, it assumed only a point mass representation of the missile. The 6DOF simulation had wind tunnel test data used in the calculation of aerodynamic forces and moments, and could be used to examine performance under more realistic conditions.

The original work [2] assumed no lag in the autopilot, another possible source of trouble. A more complex control law based on a first order autopilot lag model had been derived [9] and was available for implementation if needed. Perhaps the most crucial assumption of all to be tested, was that of no hard constraint on the controller as is present on the missile in the tail fin stops. The only constraint in the formulation of the control problem was a 'soft constraint' present in the cost function J to be minimized, Equation (3.3).

The approach used was to implement the control law with the set of state variables given in [2]: missile attitude angle θ, projected missile-to-target distance on ground Y_d and rate $\dot{Y}_d$. Perfect knowledge of the states were assumed. Since Y_d, $\dot{Y}_d$ cannot be measured, the set of state variables was then altered to include the line of sight angle λ and rate $\dot{\lambda}$. In this setting, all variables were to be assumed available at first, and then realistic measurements from the seeker for $\lambda, \dot{\lambda}$ was to be used, along with a gyro measured attitude angle θ. See section 3.2 for a more complete outline of the approaches used, along with other sections of Chapters III and IV.

Conclusions and recommendations for future study appear in Chapter V.

Chapter II

IMPLEMENTATION OF THE 6DOF SIMULATION ON THE IBM370 SYSTEM

2.1 Incompatability of the IBM, CDC FORTRAN's

One might be inclined to think that all FORTRAN's are compatable, but such is certainly not the case. Each computing system, whether it be that of IBM or CDC, has certain unique features of which a programmer can take advantage. The code of the 6DOF simulation took extensive advantage of such features and made implementation on the IBM370 system quite difficult. All incompatabilities between the two FORTRAN's had to be detected and resolved--a process which required several months of effort. The final goal of the altering of the FORTRAN code was to produce a simulation that would run on the IBM370 as well as the CDC6600, that is, the code was not to take advantage of the special features of either system.

Some of the programming inconsistencies were easy to spot and correct. Among these were the use of '*' for titles in FORMAT statements, variable names that were too long, restrictions on handling alphanumeric character data, restrictions on combining the initialization of data for arrays and specifying their length, the number of lines on which a statement can be continued, restrictions on the indexing of parameters in DO loops.

Other programming incompatabilities were more difficult to detect and correct. The plotting capability in the simulation made extensive use of in house plotting subroutines and had to be bypassed. A very troublesome programming practice was that of not initializing variables when their initial value was to be '0'. The CDC machine will zero out its memory with the start of any new job. The IBM computer does not. The 'garbage' in those particular memory locations will be used in the executation phase without any warning to the user. The way this problem was finally solved was to run the program with a WATFIV compiler which will flag uninitialized variables.

Some subroutines such as those involved with linear interpolation of one or two variables, made such extensive use of special CDC array manipulation features that they had to be rewritten and, consequently, retested for accuracy.

2.2 Disk Storage and CJS

The simulation was of such length that disk storage was essential. Once the simulation was placed on disk, the problem was still how to implement the numerous corrections decided upon. To this end, it was decided to use a new IBM editing creation called CJS, Conversational Job System which was a subset of a larger package called CMS, Conversational Monitor System. Especially desirable features included were the ability to search for phrases, to delete or add lines, rearrange whole sections of code, to make global changes as well as local ones, and to create executive programs to serve a wide variety of needs.

To reduce execution time, all subroutines which were to remain unchanged were stored on disk in object form to lessen compilation time.

2.3 Selection of Compiler

The WATFIV FORTRAN compiler was chosen for the initial phase of the work because of its rather extensive collection of syntax error messages. Once the syntax errors were removed, the G-level compiler was used because of its speed in compiling. When finally the point was reached were code changes were few, the H-level compiler was incorporated because it optimizes the execution of the code. In order to do this, it does take longer to compile.

2.4 Simulation Simplification for Developmental Work

For the developmental phase, a shorter, less complex simulation was desired, one that would be less expensive to run. The original 6DOF simulation provided capabilities that were neither needed nor desired. A variety of seeker modules, actuators, and autopilots were available. The simulation was set up to run Monte Carlo sets and to give various statistical analyses for the results. There was in addition a plotting capability which would be quite desirable but was unusable due to the utilization of subroutines found only on the CDC system.

To reduce compilation and run time, it was decided to pare down the simulation so that the resulting shorter form of the simulation would contain only the modules and subroutines necessary for the development work. To this end, some 27 subroutines were deleted. They were C2, C2I, NORMAL, RANNUM, MCARLO, AERROR, TERROR, CEPAS, CEPP, NORM, KSTEST, TABLE, PPLOT, XLOC, G2, G2I, S4, S4I, C5, C5I, RESET, PLOT4, S2, S3, PLOT2, PLOTN, and SUBL1.

A listing of the shortened form of the 6DOF simulation appears in Chapter VI.

2.5 Additional Capability Added

2.5.1 Roll, Yaw Control

Since the initial work in [2] assumed motion in one plane only, the capability of controlling roll and yaw motion was added. The simulation already had a roll control feature, but this had to be altered somewhat when the yaw control was added. The user sets switch OPTN3 equal to 0 or 1, depending on whether he wishes to allow the missile to roll or not, respectively. If no roll is selected, then by setting another switch OPTNYW to 0 or 1, he can

allow the missile to leave the pitch plane or force it to fly in it.

The coding of this alteration of subroutine D2 follows:

```
      IF (OPTN3.LE.0.) GO TO 45
      IF (OPTNYW.LE.0) GO TO 55
      GO TO 65
   45 WPD = CRAD*FMXBA/FMIX
   55 WRD = (CRAD*FMZBA+(FMIX-FMIY)*
              WP*WQ/CRAD)/FMIZ
   65 WQD = (CRAD*FMYBA+(FMIZ-FMIX)*
              WP*WR/CRAD)/FMIY
```

2.5.2 Plotting Capability

The original plotting capability was replaced with one provided by Dr. St. Clair which would generate line printer plots. Any variable in the C-array could be plotted against time, and any two variables in the C-array could be plotted against each other. Since the graphs are done on the line printer, they are rather 'crude' in appearance, but they do provide some insight into how variables are changing with respect to time.

The variables to be plotted are stored on disk. Although the plotting subroutines that are used on the IBM370 cannot be used on the CDC6600, any program using a package compatable with the CDC 6600 could be used to read off these stored variables.

2.6 General Review of the 6DOF Simulation

The modularization of the 6DOF digital simulation of a semiactive laser-guided air-to-ground missile is given in Figure 2.1. A block diagram for one channel (pitch or yaw) is shown in Figure 2.2, and a brief description of the various components follows. For a more detailed examination, refer to [11] and [12].

2.6.1 Airframe Model

The airframe model included translational and rotational dynamics and generation of aerodynamic forces and moments. Airframe dynamics were simulated in 6-DOF with the missile orientation represented by three Euler angles. Maneuvering control was achieved by means by four surfaces in cruciform configuration hinged about fixed stabilizing fins.

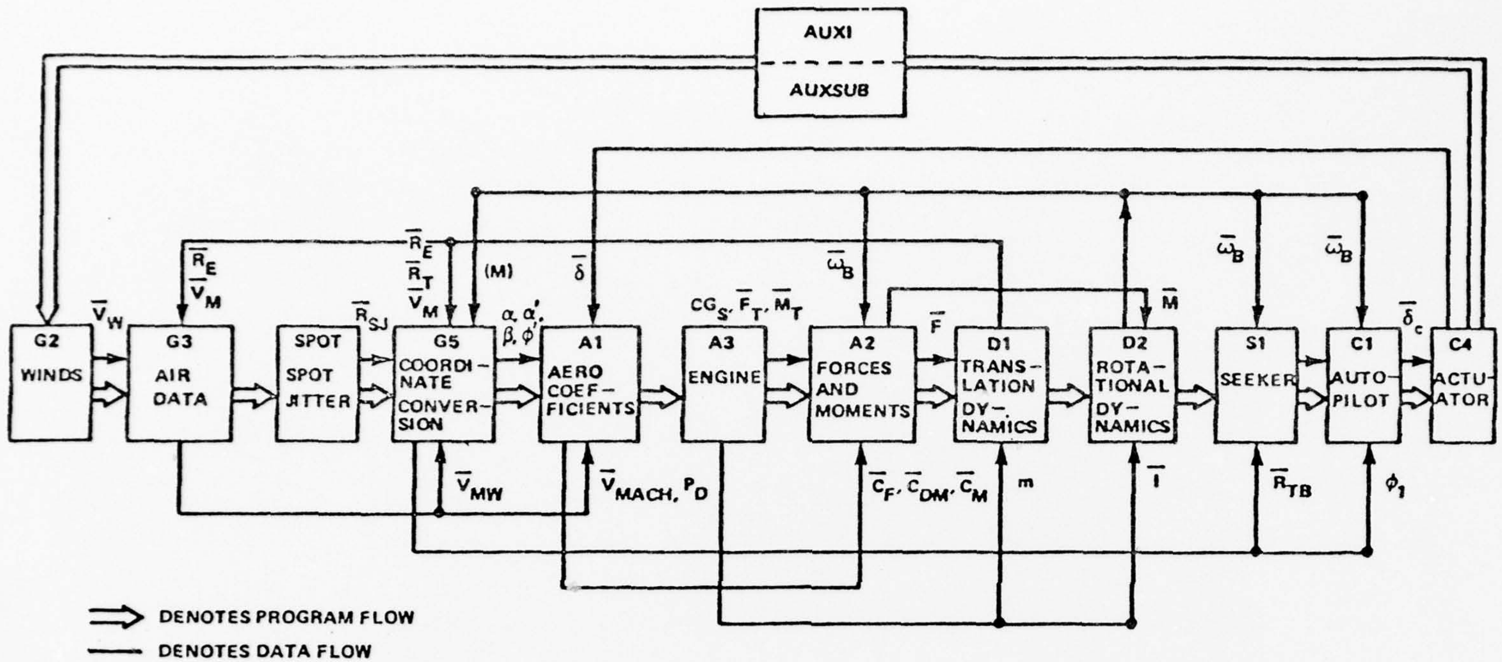

Figure 2.1. Modularization of the 6DOF Simulation

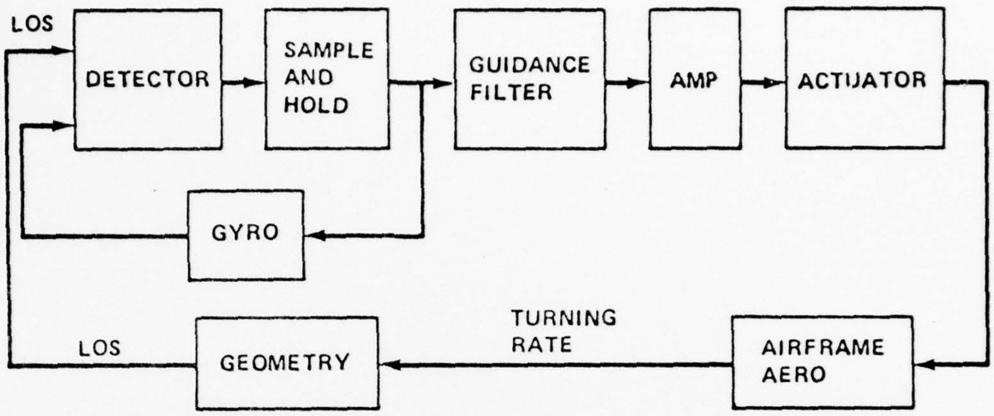

Figure 2.2. Block diagram of pitch or yaw guidance channel.

2.6.2 Rotation

$$I_X\dot{P}_B = M_{XA} \tag{2.1}$$

$$I_Y\dot{Q}_B = M_{YA} + (I_Y - I_X)P_BR_B \tag{2.2}$$

$$I_Y\dot{R}_B = M_{ZA} - (I_Y - I_X)P_BQ_B \tag{2.3}$$

where P_B, Q_B, R_B are rotational rate components about body axes X_B, Y_B, Z_B respectively; I_X and I_Y are moments of inertia about X_B and Y_B; and M_{XA}, M_{YA}, M_{ZA} are components of aerodynamic moment about the body axes.

By resolving body rotational rates into the inertial and intermediate axis systems, the following equations for Euler angles are obtained:

$$\dot{\theta}_E = (Q_B\cos\phi_E - R_B\sin\phi_E)/\cos\psi_E \tag{2.4}$$

$$\dot{\psi}_E = Q_B\sin\phi_E + R_B\cos\phi_E \tag{2.5}$$

$$\dot{\phi}_E = P_B - \dot{\theta}_E\sin\psi_E \quad . \tag{2.6}$$

2.6.3 Translation

It is advantageous to express the translation equations of motion in terms of inertial axes giving

$$m\ddot{X} = F_{XA} \tag{2.7}$$

$$m\ddot{Y} = F_{YA} \tag{2.8}$$

$$m\ddot{Z} = F_{ZA} - mg \tag{2.9}$$

where m is the mass of the missile; F_{XA}, F_{YA}, F_{ZA} are the aerodynamic forces in the inertial axes; and g is acceleration due to gravity. Aerodynamic forces are obtained in inertial axis form by transforming body axis components according to the following matrix equation:

$$\begin{bmatrix} F_{XA} \\ F_{YA} \\ F_{ZA} \end{bmatrix} = [T] \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} \qquad (2.10)$$

where F_X, F_Y, F_Z are the body axis aerodynamic force components. The body-earth transformation matrix has elements defined in [8].

2.6.4 Aerodynamic Forces and Moments

For both missile models, aerodynamic forces and moments are expressed in terms of body axes. The missile trajectory is divided into two sections; unguided flight before the target laser spot is visible (the preacquisition phase) and the guided portion of the trajectory after the seeker has acquired the target spot (the post-acquisition phase).

Aerodynamic force and moment components for both flight phases are as follows:

$$F_X = -qSC_{DT} \qquad (2.11)$$

$$F_Y = qSC_Y \qquad (2.12)$$

$$F_Z = qSC_N \qquad (2.13)$$

$$M_{XA} = qSD(C_D + DP_B C_P/2V) \qquad (2.14)$$

$$M_{YA} = qSD(C_{MCG} + DQ_B C_{MQ}/2V) \qquad (2.15)$$

$$M_{ZA} = qSD(-C_{NCG} + DR_B C_{NR}/2V) \qquad (2.16)$$

where q is the dynamic pressure, S is a reference area (cross-sectional area of the missile), D is a reference length (diameter of the missile), and V is the total speed of the missile relative to the atmosphere. Dynamic pressure is given by

$$q = pV^2/2 \qquad (2.17)$$

where p is atmospheric density and is a function of altitude.

Aerodynamic coefficients and derivatives referenced in Equations (2.11) through (2.16) are obtained from wind tunnel tests and other estimates are expressed in graphical form as functions of angle of attack, Mach number and, during the post-acquisition phase, control vane angles. Coefficients and derivatives are defined for both models by differing functions for the two flight phases. Details of the coefficients for both models are presented in [8].

Angle of attack components (α and β) may be defined by the body axis components of velocity. However, for digital computation, there is some advantage in expressing α and β in terms of integral equations using wind axis coordinates. Thus, using equations for $\dot{\alpha}$ and $\dot{\beta}$ derived in [8] and making small angle approximations, the equations for angle of attack components are:

$$\dot{\alpha} = (F_{ZB} - \alpha F_{XB})/mV - \beta P_B + Q_B \tag{2.18}$$

$$\dot{\beta} = (F_{YB} - \beta F_{XB})/mV + \alpha P_B - R_B \tag{2.19}$$

where the body force components (F_{XB}, F_{YB}, F_{ZB}) include aerodynamic forces and gravity force resolved into body axis components.

2.6.5 Target Model

The target model included target position as a function of time and the calculation of missile-target miss distance at intercept. Target motion was specified deterministically as either constant acceleration or constant velocity from an initial starting point. However, the effect of "jitter" in the designating laser beam direction was introduced by applying a random disturbance to the calculated target position.

Target to missile displacements are

$$\Delta X = X_t - X_m \tag{2.20}$$

$$\Delta Y = Y_t - Y_m \tag{2.21}$$

$$\Delta Z = Z_t - Z_m \tag{2.22}$$

where X_t, Y_t, Z_t are target position coordinates and X_m, T_m, Z_m are missile position coordinates. The gyro platform is characterized by two Euler angles (θ_{SI}, ψ_{SI}) which define the orientation of the platform axes relative to inertial axes parallel to X, Y, Z. Target to missile displacements in digital seeker axes are given by

$$\begin{bmatrix} \Delta X_S \\ \Delta Y_S \\ \Delta Z_S \end{bmatrix} = [S] \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} \tag{2.23}$$

where the transformation matrix [S] has elements as shown in [8].

From the target to missile displacements in seeker coordinates, the boresight error angles are defined as:

$$\text{BEPSZ} = \tan^{-1} (-\Delta Z_S/\Delta X_S) \tag{2.24}$$

$$\text{BEPSY} = \tan^{-1}(\Delta Y_S/ \sqrt{\Delta X_S^2 + \Delta Z_S^2}) \quad . \tag{2.25}$$

2.6.6 Guidance and Control Models

The guidance and control models included the seeker detector and gyro, guidance filters, and vane actuators. The laser detector is mounted on a 2-DOF gyro-stabilized platform in the nose of the missile, and the laser beam is viewed through a fixed lens. After the gyro has been uncaged during the terminal homing phase of the missile flight, gimbal torquing signals are generated to null the spot displacement from the center of the detector and the resulting gimbal angles and rates relative to the missile are used to generate guidance and stabilizing signals for input to the guidance filters. Detector output is from a sample-and-hold unit which operates at a fixed period.

A proportional navigation guidance scheme is employed; the guidance filters are lead-lag networks in each of the pitch and yaw channels which serve to decouple the missile natural frequency in pitch and yaw from the control system.

The ideal characteristics of the seeker transfer function are shown in Figure 2.3.

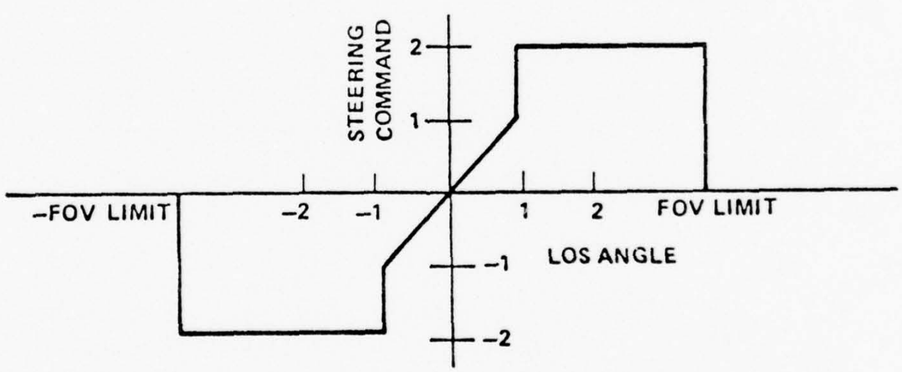

Figure 2.3. Ideal seeker characteristics.

2.6.7 Gyroscope Model

For many of the studies using the digital simulation, it is desirable to model a gyro which has no dynamic time constants. The simplest model possible is one represented by a perfect integration, i.e., 1/s in the Laplace notation.

As is well known [8], for example, the differential equation relating output axis motion to input torque or input rate is given by

$$J\ddot{\theta} + B\dot{\theta} + K\theta = -H\dot{\phi}$$

where $J \rightarrow$ ft-lb/(rad/sec^2), $B \rightarrow$ ft-lb/(rad/sec), $K \rightarrow$ ft/rad, and $H \rightarrow$ ft-lb/(rad/sec).

2.6.8 Guidance Filter Model

The guidance filter or compensation network is of the lead-lag type with transfer function $(\tau_3 s + 1)/(\tau_4 s + 1)$, where τ_3 is the lead time constant and τ_4 is the lag time constant. The input to the guidance filter comes from two sources, the sample-and-hold output of the seeker and the damping network output.

2.6.9 Actuator Model

A first-order actuator was used in the simulation with transfer function AA/(s + AA) where AA is the actuator model constant. The input to the actuator is from the guidance filter, and the output drives the canards. The yaw control canards are on a common shaft, as are the pitch canards.

Chapter III

CONTROL LAW IMPLEMENTATION - IDEAL CASE

3.1 Brief Summary of Control Law

In the paper "Terminal Guidance for Impact Attitude Angle Constrained Flight Trajectories" [2], a simplified model of a missile - target scenario is used to produce a three state control law that will result in the missile impacting vertically on the target, Figure 3.1 deplicts the geometry of the terminal guidance phase, and Table 3.1 defines the variables introduced.

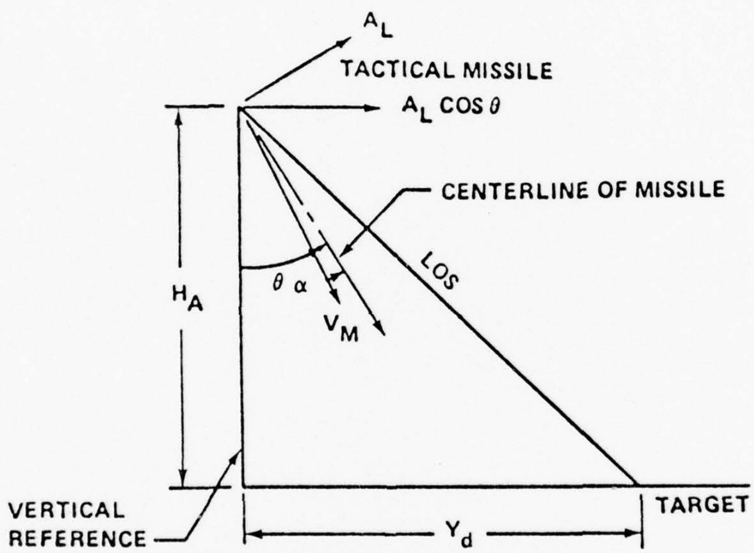

Figure 3.1. Geometry of Tactical Missile - Target Positions.

Table 3.1. Definition of Variables

Variable	Definition
Y_m	Missile position variable projected on the ground (ft)
Y_t	Target position variable (ft)
Y_d	Position variable from missile to target projected on the ground ($Y_d = Y_t - Y_m$)

Table 3.1. Continued

Variable	Definition
$\dot{Y}_d$	Time derivative of Y_d (ft/sec)
A_L	Lateral acceleration of the missile (ft/sec^2)
θ	Body attitude angle of the missile (deg)

The following set of state variables was chosen for the modeling process.

$$\bar{X} = \begin{bmatrix} Y_d \\ \dot{Y}_d \\ A_L \\ \theta \end{bmatrix} . \tag{3.1}$$

Under the following assumptions:

1) the angle of attack is small and can thus be neglected, and

2) the autopilot has zero lag,

$$u = c_1(t)Y_d + c_2(t)\dot{Y}_d + c_3(t)\theta \tag{3.2}$$

a control law of the form can be obtained which will minimize the performance index

$$J = Y_d^{\,2}(t_f) + \gamma\theta^2(t_f) + \beta\int_{t_o}^{t_f} u^2(t)dt \tag{3.3}$$

where the state variables are subject to the following dynamics:

$$\dot{Y}_d = \dot{Y}_d \tag{3.4}$$

$$\ddot{Y}_d = -A_L \cos\theta \tag{3.5}$$

$$\dot{A}_L = -w_1 A_L + K_1 u \tag{3.6}$$

$$\dot{\theta} = K_a u \quad . \tag{3.7}$$

Performance is considered to be acceptable when the following constraints are satisfied:

$$|Y_d(t_f)| \leq 5 \text{ feet} \tag{3.8}$$

$$|\theta(t_f)| \leq 5 \text{ degrees} . \tag{3.9}$$

The time varying coefficients appearing in the control law are given by:

$$c_1 = [-\beta g(t_f-t) - g\gamma K_a^2(t_f-t)^2/2]/\Delta \tag{3.10}$$

$$c_2 = [-\beta g(t_f-t)^2 - g\gamma K_a^2(t_f-t)^3/2]/\Delta \tag{3.11}$$

$$c_3 = [-\beta\gamma K_a + \gamma K_a g^2(t_f-t)^3/6]/\Delta \tag{3.12}$$

where

$$g = \frac{-bK_1}{w_1} \quad (b = \cos\theta \text{ in the linearized form}) \tag{3.13}$$

and

$$\Delta \equiv \beta^2 + \gamma\beta K_a^2(t_f-t) + \beta g^2(t_f-t)^3/3 + \gamma g^2 K_a^2(t_f-t)^4/12, \tag{3.14}$$

t = time, t_f = time of impact.

A derivation of this control law is given in Appendix E.

The assumption of no lag in the autopilot has been removed by Pastrick and York, "Optimal Terminal Guidance with Constraints at Final Time" [9]. This work discusses a control law more general in nature which allows the control law of [2] as a limiting case. Appendix A contains a detailed examination of this more realistic, and consequently, more complex control law.

3.2 General Outline of Approach

A simplified simulation using the dynamics as modeled in section 3.2 was developed (Appendix B). Whenever feasable, new ideas were tried out on the simplified simulation first to test their merit. This approach is not only philosophically sound, but there was a very practical reason for it as well--money, or rather the lack thereof. The large scale simulation received from Redstone Arsenal required large amounts of memory and was rather expensive to run (@ \$20 for a complete flight).

The smaller four state simulation was made to resemble the large 6DOF one as closely as possible. The original constant velocity was replaced with a time varying velocity profile taken from a 6DOF simulation run. The parameters K_1, K_a, w_1 that appear in the four state dynamics were re-examined and evaluated to be sure that they reflected the dynamics of the actual weapon system. It was found that the parameter values given in [2] of a typical tactical missile did reflect the dynamics in our case as well. One discrepancy was that our autopilot lag was more accurately represented by

$$w_1 = 9.8 \tag{3.15}$$

instead of

$$w_1 = 5, \tag{3.16}$$

but it was decided to use the smaller value in the developmental work as this represents more lag in autopilot. Since our small scale simulation ignores aerodynamic forces and moments, it was felt that this increased lag might make our simple simulation even more representative of the actual system.

The 6DOF simulation was altered as well. Since the initial work in [2] assumed flight in one plane only, it was decided to first try to implement just the pitch control, and then later a yaw control. Consequently, the roll and yaw control parameters were set so that the missile was allowed to neither roll or yaw. Once the control law was properly implemented in the pitch channel, then symmetry would suggest that it could be implemented in the same manner in the yaw channel.

Once the control law was performing properly with the four state simulation, then it would be used on the 6DOF simulation. The approach used in implementing the control law was to assume that all of the states Y_d, $\dot{Y}_d$, and θ were known and could be measured exactly, and that it was possible to generate the time varying coefficients c_1, c_2, c_3 without error. It was felt that for the initial implementation that it was better to leave the choice of states as given in [2], although there is in fact no way to measure the range and range rate states of Y_d and $\dot{Y}_d$. For this reason, this implementation is referred to as the ideal case. For the practical case, an alternative set of state variables was used.

The control parameter values appearing in [2] were optimized for a vertical impact angle, but the initial geometry was that of a forty-five degree triangle (10,000' from the target - 10,000' up). When using their control parameter values of

$$\gamma = 3823 \qquad \text{and} \qquad \beta = 6.94\text{E-}04 \tag{3.17}$$

with our low profile trajectory (200' x 2600'), the missile in the simple simulation would overfly the target. The same type of trajectory was obtained on the 6DOF simulation. Basically, the trouble seemed to be that the missile could not turn in the time allowed. The presence of aerodynamic forces and moments made this problem even more acute.

It was decided to first try to achieve an attitude angle that would be less demanding on the control system, such as

$$\theta(t_f) = 45 \text{ degrees.} \tag{3.18}$$

Appendix A shows how it is possible to use the same control law to achieve an arbitrary angle. These initial runs also suggested that the geometry for the terminal guidance phase needed to be altered to allow more height with which to work. Thus, it was decided that the trajectory would need to consist of two phases: a pre-programmed pitch maneuver to gain altitude followed by a terminal guidance phase.

Initial conditions for the terminal guidance phase were chosen to be:

$$H_t = 1000', \; Y_d = 5000', \; \theta = 85^{\circ} . \tag{3.19}$$

The same assumptions were made as in [2] of no angle of attack and no lag in the autopilot. It was decided to implement the 3-state controller because of its relatively simple form. Should lag become a problem, the 4-state controller discussed in Appendix A could be tried. The basic problem then was to find the optimal values of the two control parameters γ and β that will minimize the cost functional J given as Equation (3.3). This problem was solved using the Hooke - Jeeves Algorithm as discussed in section 3.3.

With the following parameter values,

$$\gamma = 5525.51 \qquad \text{and} \qquad \beta = .19E\text{-}06 \tag{3.20}$$

a miss distance of 3.3' was achieved with an attitude angle at impact of 45.27°. With these control parameter values, the control law was used with the 6DOF simulation. Section 3.4 discusses how the control law was implemented. In summary, though, a miss distance of 18 feet and an attitude angle of 37 degrees was obtained before efforts were made to improve on the control parameter values, see section 3.4. Finally, performance sensitivity to initial conditions is discussed in section 3.5.

3.3 Determination of Control Parameters γ and β

Two control parameters appear in the cost functional to be minimized:

$$J = Y_d^{\,2}(t_f) + \gamma[\theta(t_f)-45]^2 + \beta\int_{t_o}^{t_f} u^2(t)dt \quad . \tag{3.21}$$

The problem was to determine optimal values for γ and β so that the following success criteria could be achieved:

$$|Y_d(t_f)| \leq 5 \text{ feet} \tag{3.22}$$

$$|\theta(t_f)-45^{\circ}| \leq 5 \text{ degrees} . \tag{3.23}$$

The approach used to solve this problem was to view it as the mathematical programming problem of minimizing

$$F(\gamma,\beta) = Y_d^{\,2}(t_f) + [\theta(t_f)-45]^2 \tag{3.24}$$

where the function evaluation was achieved through a computer run of the four-state simulation program with specified values for γ and β. (see Appendix C for a discussion of mathematical programming in general and this problem in particular).

To make the mathematical programming problem mathematically feasible, it was assumed that $F(\gamma,\beta)$ varied in a continuous manner with respect to its two arguments. Since $F(\gamma,\beta)$ could not be expressed in any closed form, there was no derivative information available. After some thought it was decided to use the Hooke - Jeeves Algorithm to tackle this problem as it is rather straight forward, has almost guaranteed convergence, and was readily available.

This particular algorithm will make two exploratory search moves, one in each coordinate direction, to reduce $F(\gamma,\beta)$, and then will follow this with a pattern search (in the direction of the diagonal).

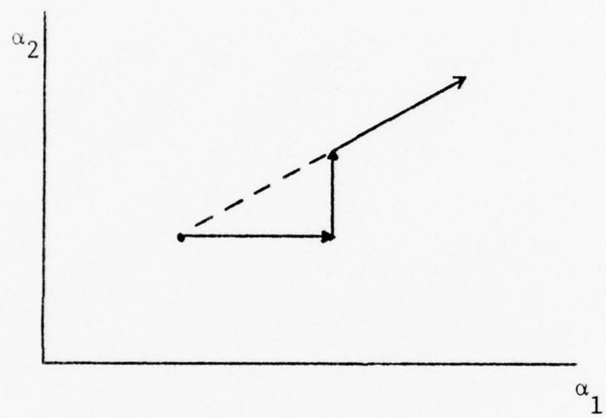

Figure 3.2. Hooke - Jeeves Algorithm for n = 2

Using this algorithm, the optimal control parameter values were found to be

$$\gamma = 5525.508 \qquad \beta = 190E-06 \tag{3.25}$$

which yielded the following results:

$$Y_d(t_f) = 3.3' \quad \text{and} \quad \theta(t_f) = 45.27 \ . \tag{3.26}$$

3.4 6DOF Simulation Implementation

The first task in implementing the three-state optimal control law was to modify the control module C1. This modification is given in section 3.4.1.

The second task was to evaluate the two gains GPIT, GYAW that appear in the control module, see section 3.4.2.

The last task was to improve upon the values of γ and β given by the Hooke - Jeeves Algorithm working with the four-state simulation which was being used to approximate the 6DOF simulation. This improvement is discussed in section 3.4.3.

3.4.1 Modification of Control Module C1

3.4.1a. Implementation of the Control Law

Figure 3.3 gives the original feedback control system based on proportional navigation. As discussed in section 3.1, it was decided to initially control pitch motion only. The roll stabilization channel was left in tact, as was the cross coupling of the pitch and yaw control channels. This cross coupling is necessary since this particular missile when it is roll stabilized flies with the following tail fin configuration:

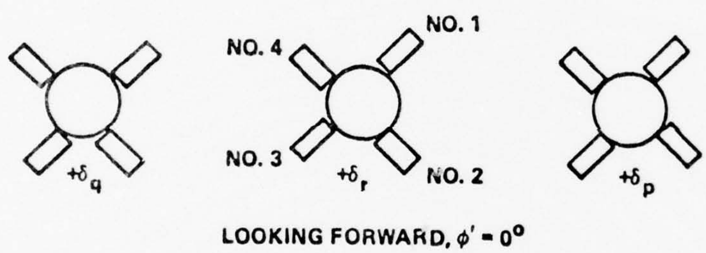

Figure 3.4. Missile Roll Stabilized Position

All four tail fins are required to execute a pitch maneuver, as would also be true for a yaw or roll maneuver.

The initial command signal to the tail fin occurs at the first summation in the pitch (upward) and yaw (lower) control channels. This signal is the difference between the commanded attitude angle and the sensed attitude angle (the feedback signal). The new commanded attitude angle was given by the control law

$$u = c_1(t)Y_d + c_2(t)\dot{Y}_d + c_3(t)\theta \, . \qquad (3.27)$$

The guidance filters in the pitch and yaw channels were replaced by gains GPIT and GYAW for the initial implementation. The new autopilot is given in Figure 3.5.

3.4.1b. Control Module C1 Listing

The new variables that have been introduced are given in Table 3.2

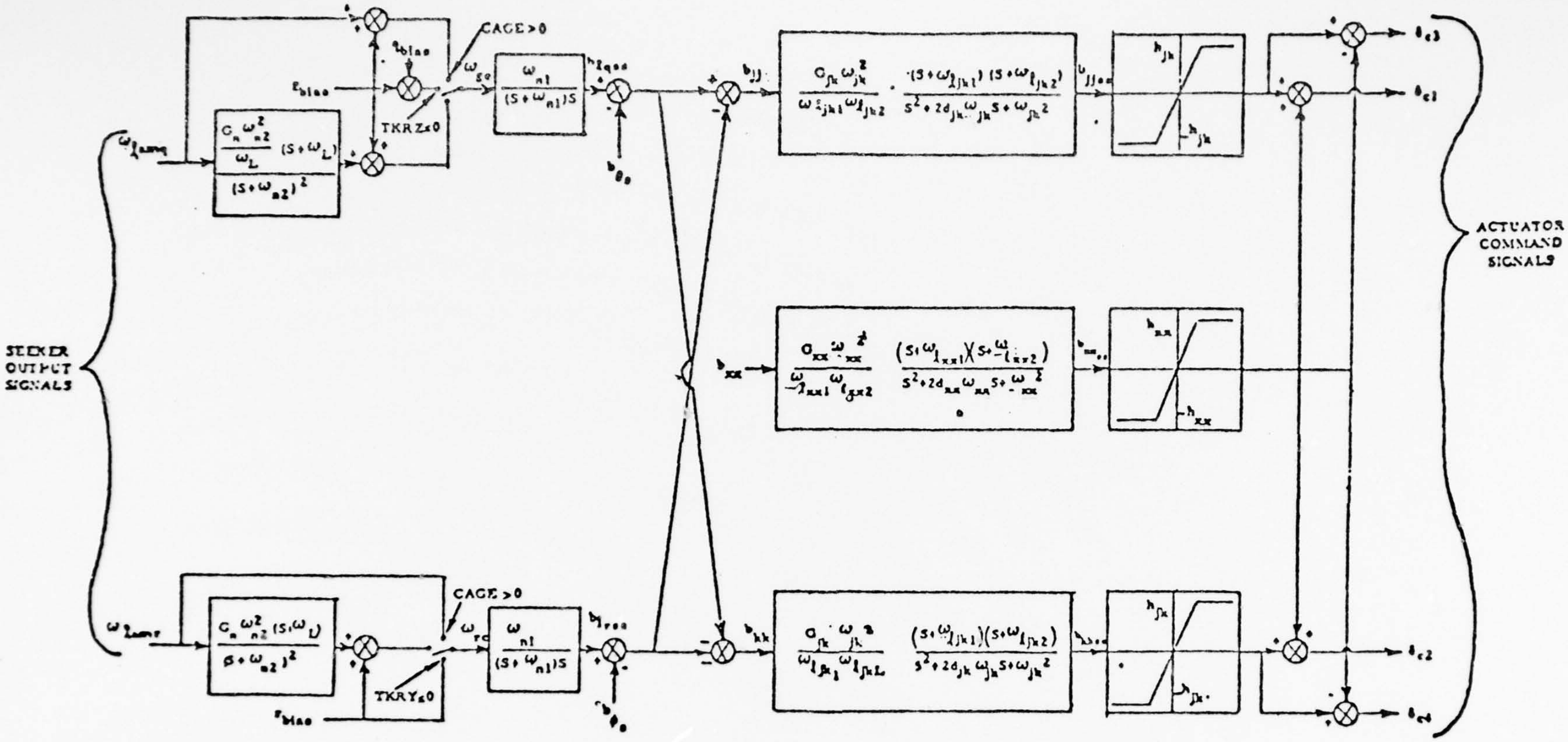

Figure 3.3. Original Feedback Control System

Table 3.2. New Variables for Subroutine C1

Name	Symbol	Location in C Array	Definition
BETA	β	2917	Control parameter
BDELTC(I)	δ_{ci}	856	Commanded tail fin position, i = 1,2,3,4
BJJ		881	Pitch signal after coupling
BJS		537	Pitch signal before coupling
BKK		882	Yaw signal after coupling
BKS		545	Yaw signal before coupling
BJJSSS		987	Pitch signal before limiter
BKKSSS		988	Yaw signal before limiter
BTHT		350	Attitude pitch angle (measured from horizon)
BXX			Roll signal before network
BXXSSS		989	Roll signal after limiter
CTHTA(T)	$c_3(t)$		Control law coefficient for θ
CXREL(T)	$c_1(t)$		Control law coefficient for Y_d
CXRELD(T)	$c_2(t)$		Control law coefficient for $\dot{Y}_d$
DELTA(T)	Δ		Denominator for control coefficients
G	g		$-bK_1/w_1$
GAMMA	γ	2916	Control parameter
GPIT		2921	Gain in pitch channel
GYAW		2922	Gain in yaw channel
KA	K_a	2918	Proportionality constant appearing in dynamics
K1	K_1	2919	Constant appearing in autopilot lag model
T	t	2000	Time
TGOX			Estimate of time-to-go in x direction
TGOY			Estimate of time-to-go in z direction

Table 3.2. Continued

Name	Symbol	Location in C Array	Description
TGO			Min (TGOX, TGOY)
TF	t_f	2906	Estimate of time of impact
W1	w_1	2920	Autopilot lag constant
XREL	Y_d	2904	Relative x-direction missile - target position (projected on the ground)
XRELD	$\dot{Y}_d$	2905	Time derivative of XREL
YREL		2902	Relative y-direction missile - target position (projected on the ground)
YRELD		2903	Time derivative of YREL

```
      SUBROUTINE C1
      COMMON C(3830),GRAPH(300,4)
      DIMENSION BDELTC(4),VAR(101)
C
C**INPUT DATA
      EQUIVALENCE (C( 860),TDY    )
      EQUIVALENCE (C( 861),GBIAS  )
      EQUIVALENCE (C( 862),GN     )
      EQUIVALENCE (C( 863),WN2    )
      EQUIVALENCE (C( 864),WN1    )
      EQUIVALENCE (C( 865),WL     )
      EQUIVALENCE (C( 866),WLXX1  )
      EQUIVALENCE (C( 867),WLXX2  )
      EQUIVALENCE (C( 868),WLJK1  )
      EQUIVALENCE (C( 869),WLJK2  )
      EQUIVALENCE (C( 870),HJK    )
      EQUIVALENCE (C( 871),WXX    )
      EQUIVALENCE (C( 872),DXX    )
      EQUIVALENCE (C( 873),WJK    )
      EQUIVALENCE (C( 874),DJK    )
      EQUIVALENCE (C( 875),GXX    )
      EQUIVALENCE (C( 876),GJK    )
      EQUIVALENCE (C( 877),RES    )
      EQUIVALENCE (C( 878),QDN    )
      EQUIVALENCE (C( 879),QUP    )
      EQUIVALENCE (C( 890),HXX    )
      EQUIVALENCE (C( 892),QBIAS  )
      EQUIVALENCE (C( 893),RBIAS  )
      EQUIVALENCE (C( 899),OPTC1  )
      EQUIVALENCE (C( 947),GNS    )
      EQUIVALENCE (C( 948),WS1    )
      EQUIVALENCE (C( 949),WS2    )
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(  77),SPHI    )
      EQUIVALENCE (C(  87),STHT    )
      EQUIVALENCE (C(  97),SPSI    )
      EQUIVALENCE (C( 353),BPH1   )
      EQUIVALENCE (C( 354),BTH2   )
      EQUIVALENCE (C( 355),BPS1   )
      EQUIVALENCE (C( 403),WLAMQ  )
      EQUIVALENCE (C( 407),WLAMR  )
      EQUIVALENCE (C( 461),CAGE   )
      EQUIVALENCE (C( 462),TKRZ   )
      EQUIVALENCE (C( 463),TKRY   )
      EQUIVALENCE (C(1233),BDR    )
      EQUIVALENCE (C(1747),WR     )
      EQUIVALENCE (C(1743),WQ     )
      EQUIVALENCE (C(1739),WP     )
C
C**INPUTS FROM MAIN PROGRAM
      EQUIVALENCE (C(2000),T      )
C
C** STATE VARIABLE OUTPUTS
      EQUIVALENCE (C( 800),WLQSDD)
      EQUIVALENCE (C( 803),WLQSP )
      EQUIVALENCE (C( 804),WLQSD )
      EQUIVALENCE (C( 807),WLQS  )
      EQUIVALENCE (C( 808),WLQSSD)
```

```
      EQUIVALENCE (C( 811),WLQSS )
      EQUIVALENCE (C( 812),WLRSDD)
      EQUIVALENCE (C( 815),WLRSP )
      EQUIVALENCE (C( 816),WLRSD )
      EQUIVALENCE (C( 819),WLRS  )
      EQUIVALENCE (C( 820),WLRSSD)
      EQUIVALENCE (C( 823),WLRSS )
      EQUIVALENCE (C( 824),BLQSSD)
      EQUIVALENCE (C( 827),BLQSS )
      EQUIVALENCE (C( 828),BLRSSD)
      EQUIVALENCE (C( 831),BLRSS )
      EQUIVALENCE (C( 832),BJJSDD)
      EQUIVALENCE (C( 835),BJJSP )
      EQUIVALENCE (C( 836),BJJSD )
      EQUIVALENCE (C( 839),BJJS  )
      EQUIVALENCE (C( 840),BKKSDD)
      EQUIVALENCE (C( 843),BKKSP )
      EQUIVALENCE (C( 844),BKKSD )
      EQUIVALENCE (C( 847),BKKS  )
      EQUIVALENCE (C( 848),BXXSDD)
      EQUIVALENCE (C( 851),BXXSP )
      EQUIVALENCE (C( 852),BXXSD )
      EQUIVALENCE (C( 855),BXXS  )
      EQUIVALENCE (C( 931),BJSSD ), (C( 934),BJSS  )
      EQUIVALENCE (C( 935),BKSSD ), (C( 938),BKSS  )
      EQUIVALENCE (C( 950),SNP2  ), (C( 953),SNP1  ),(C( 956),SNP0  )
      EQUIVALENCE (C( 957),SNQ2  ), (C( 960),SNQ1  ),(C( 963),SNQ0  )
      EQUIVALENCE (C( 964),SNR2  ), (C( 967),SNR1  ),(C( 970),SNR0  )
      EQUIVALENCE (C( 971),BPC2  ), (C( 974),BPC1  ),(C( 977),BPC0  )
      EQUIVALENCE (C(903),H13P),(C(904),H13M)
      EQUIVALENCE (C(905),H24P),(C(906),H24M)
      EQUIVALENCE (C(907),CDRFT1),(C(908),CDRFT2)
      EQUIVALENCE (C(909),CDRFTY)
      EQUIVALENCE (C(984),CDRFTX)
      EQUIVALENCE (C(1676),ANGX)
      EQUIVALENCE (C(978),BDRFTD),(C(981),BDRFT)
      EQUIVALENCE (C(985),NLMT1),(C(986),NLMT2)
      EQUIVALENCE (C(987),BJJSSS),(C(988),BKKSSS)
      EQUIVALENCE (C(989),BXXSSS)
      EQUIVALENCE (C(990),BJJSSL),(C(991),BKKSSL)
      EQUIVALENCE (C(530),BJSDD),(C(534),BJSD),(C(537),BJS)
      EQUIVALENCE (C(538),BKSDD),(C(542),BKSD),(C(545),BKS)
      EQUIVALENCE (C(546),BXSDD),(C(550),BXSD),(C(553),BXS)
      EQUIVALENCE (C(533),CJSD),(C(541),CKSD),(C(549),CXSD)
       EQUIVALENCE (C(942),WL2)
       EQUIVALENCE (C(943),DJ2)
       EQUIVALENCE (C(944),WJ2)
       EQUIVALENCE (C(945),DX2)
       EQUIVALENCE (C(946),WX2)
      EQUIVALENCE (C(2965),VAR(1))
C
C**OUTPUTS
      EQUIVALENCE (C( 856),BDELTC(1))
C
CM*OTHER OUTPUTS
      EQUIVALENCE (C( 880),BPHIS )
      EQUIVALENCE (C(518),B13SS)
      EQUIVALENCE (C(519),B24SS)
      EQUIVALENCE (C( 881),BJJ   )
      EQUIVALENCE (C( 882),BKK   )
```

```
      EQUIVALENCE (C( 883),BXXSS )
      EQUIVALENCE (C( 884),BJJSS )
      EQUIVALENCE (C( 885),BKKSS )
      EQUIVALENCE (C( 886),BTHTS )
      EQUIVALENCE (C( 887),BPSIS )
C**ADDITIONAL INPUT FOR NEW CONTROL LAW
      EQUIVALENCE (C(2000),T      )
      EQUIVALENCE (C( 350),BTHT   )
      EQUIVALENCE (C( 351),BPSI   )
      EQUIVALENCE (C(1615),RXE    )
      EQUIVALENCE (C(1603),VXE    )
      EQUIVALENCE (C(1619),RYE    )
      EQUIVALENCE (C(1623),RZE    )
      EQUIVALENCE (C(1611),VZE  )
      EQUIVALENCE (C(1607),VYE    )
      EQUIVALENCE (C(1651),RTXE   )
      EQUIVALENCE (C(1660),VTXE   )
      EQUIVALENCE (C(1655),RTYE   )
      EQUIVALENCE (C(1661),VTYE   )
      EQUIVALENCE (C(2916),GAMMA)
      EQUIVALENCE (C(2917),BETA)
      EQUIVALENCE (C(2918),KA)
      EQUIVALENCE (C(2919),K1)
      EQUIVALENCE (C(2920),W1)
      EQUIVALENCE (C(2921),GPIT)
      EQUIVALENCE (C(2922),GYAW)
C**ADDITIONAL OUTPUTS
      EQUIVALENCE (C(2904),XREL   )
      EQUIVALENCE (C(2905),XRELD  )
      EQUIVALENCE (C(2902),YREL   )
      EQUIVALENCE (C(2903),YRELD  )
      EQUIVALENCE(C(2906),TF)
C
      REAL KA,K1
      DATA B/.7071/
      DATA PID2/1.570796/
C
      DELTA(T)=BETA**2+GAMMA*BETA*KA**2*(TF-T)+BETA*G**2*(TF-T)**3/3.+
     .   (GAMMA*G**2*KA**2*(TF-T)**4)/12.
      CTHTA(T)=(-BETA*GAMMA*KA+(GAMMA*KA*G**2*(TF-T)**3)/6.)/DELTA(T)
      CXREL(T)=(-BETA*G*(TF-T)-(G*GAMMA*KA**2*(TF-T)**2)/2.)/DELTA(T)
      CXRELD(T)=(-BETA*G*(TF-T)**2-G*GAMMA*KA**2*(TF-T)**3/2.)/DELTA(T)
C
C**GUIDANCE SIGNAL SHAPING
C**GUIDANCE SWITCHING
      WLQSD = WLQSP
      WLRSD = WLRSP
      WLQSDD = WN2*(WN2*(WLAMQ - WLQS) - 2.*WLQSD)
      WLRSDD = WN2*(WN2*(WLAMR - WLRS) - 2.*WLRSD)
      WQC = GN*(WLQSD/WL + WLQS) + QBIAS + GBIAS
      WRC = GN*(WLRSD/WL+WLRS) + RBIAS
      IF (TKRZ.GT.0. .AND. T.GT.TDY) GO TO 4
      WLQSDD = 0.
      WQC = QBIAS + GBIAS + QDN
      IF(CAGE .GT. 0. .AND. T.GT. TDY) WQC = WLAMQ + QBIAS + GBIAS
    4 IF (TKRY .GT. 0.) GO TO 5
      WLRSDD = 0.
      WRC = RBIAS
      IF(CAGE .GT. 0.) WRC = WLAMR + RBIAS
    5 CONTINUE
```

```
      WLQSSD = WN1*(WQC - WLQSS)
      WLRSSD = WN1*(WRC - WLRSS)
      IF(WN1 .GT. 0.) GO TO 3
      WLQSS = WQC
      WLRSS = WRC
    3 BLQSSD = WLQSS
      BLRSSD = WLRSS
C
C**RATE GYRO DYNAMICS AND LIMITING
      BDRFTD=(CDRFT1*ANGX+CDRFT2)
      BTHTS=-BTH2+BDRFT
      BPSIS=-BPS1+CDRFTY*BDRFT
      BPHIS=-BPH1+CDRFTX*BDRFT
      BPHISD = WP - (WQ*COSD(-BPH1) -WR*SIND(-BPH1))
     *        *SIND(-BPS1)/COSD(-BPS1)
      BPHS = BPHISD / WLXX2 + BPHIS
    6 IF(GNS .LE. 0.) GO TO 8
      SNP2 = WS1*WS2*(GNS*SPHI-SNP0) - (WS1+WS2)*SNP1
      SNQ2 = WS1*WS2*(GNS*STHT-SNQ0) - (WS1+WS2)*SNQ1
      SNR2 = WS1*WS2*(GNS*SPSI-SNR0) - (WS1+WS2)*SNR1
      BPHS = BPHS -SNP1
      BTHTS = BTHTS - SNQ1
      BPSIS = BPSIS - SNR1
    8 CONTINUE
      BXX = BPHS
      BTSS = BTHTS
      BPSS = BPSIS
C*****SPECIAL CASE - PROGRAMMED FLIGHT
      IF(OPTC1 .LE. 0.) GO TO 9
      BLQSSD = QBIAS
      BLRSSD = 0.
      BDC = 0.
      BT=0.
      BP=0.
      IF(T.GE.1.0000 .AND. T.LE.3.2000)BDC= 5.
      IF(T.GE.4.2    .AND. T.LE.6.4000)BT= 5.
      IF(T.GE.7.6000 .AND. T.LE.9.9000)BDC=-5.
      IF(T.GE.11.000 .AND. T.LE.13.100)BP=-5.
      IF(T.GE.15.200 .AND. T.LE.17.300)BDC= 5.
      IF(T.GE.18.320 .AND. T.LE.20.560)BT = 5.
      IF(T.GE.21.545 .AND. T.LE.23.675)BDC=-5.
      IF(T.GE.24.770 .AND. T.LE.26.910)BP=-5.
      C(520)=-BPC0+BP-BT
       BPC2=18.18*20.57*(BDC-BPC0)-(18.18+20.57)*BPC1
      BXX = BXX + BPC0
      BTSS=BTSS-BT
       BPSS=BPSS-BP
C
C
    9 CONTINUE
C***NEW OPTIMAL CONTROL FOR PITCH, YAW SIGNALS
      G=-B*K1/W1
      XREL=RTXE-RXE
      YREL=RTYE-RYE
      XRELD=VTXE-VXE
      YRELD=VTYE-VYE
C****ESTIMATE OF TIME-TO-GO
      TGOX=ABS(XREL/XRELD)
   77 TGOZ=ABS(RZE/VZE)
   88 CONTINUE
```

```
      TGO= AMIN1(TGOX,TGOZ)
      TF=T+TGO
      BTHTR=BTHT/57.29578
      BJS=CTHTA(T)*(PID2+BTHTR)+CXREL(T)*XREL+CXRELD(T)*XRELD
       BJS=-BJS
      BKS=0.
      BJJ=BKS-BJS
      BKK=-BKS-BJS
      BJJSSS=GPIT*BJJ
      BKKSSS=GYAW*BKK
C
C**GUIDANCE SIGNAL SHAPING AND LIMITING
      BXXSD = BXXSP
      BXXSDD = WXX*(WXX*(BXX - BXXS) - 2.*DXX*BXXSD)
      BXSD=CXSD
      BXXSS=GXX*((BXXSDD+(WLXX1+WLXX2)*BXXSD)/(WLXX1*WLXX2)+BXXS)
      BXXSSS=BXS
C***SIGNAL LIMITING
 10   BJJSSL=BJJSSS
       BKKSSL=BKKSSS
      IF(BJJSSL.GT.H13P)BJJSSL=H13P
      IF(BJJSSL.LT.H13M)BJJSSL=H13M
      IF(BKKSSL.GT.H24P)BKKSSL=H24P
      IF(BKKSSL.LT.H24M)BKKSSL=H24M
      B13SS=BJJSSL
      B24SS=BKKSSL
C
C**COMMANDS TO ACTUATORS
      BDELTC(1)=B13SS+BXXSSS
      BDELTC(2)=B24SS+BXXSSS
      BDELTC(3)=B13SS-BXXSSS
      BDELTC(4)=B24SS-BXXSSS
      RETURN
      END
```

3.4.2 Evaluation of Control Gains GPIT, GYAW

Two new constants, GPIT and GYAW, the gains in the pitch and yaw control channels, were introduced in the control module C1 to regulate the amplitude of the tail fin command signal. As can be seen from Figure 3.5, the signal is fed through a limiter after the multiplication by the gain.

By working with four-state simulation, optimal values were obtained for the control parameters, γ and β(see section 3.3). Under the assumption that the four-state simulation approximated the 6DOF simulation, it should be possible to obtain a reasonable miss distance and impact angle with the more complex simulation by using the predetermined optimal values of the control parameters. Several runs were made with varying values of the gains GPIT and GYAW. The amplitude of the control signal had to be such that the tail fins were not at the stops throughout most of the flight, but would be hard over at the end to achieve the desired attitude angle at impact. A miss distance of 18 feet and an attitude impact angle of 37 degrees was achieved with the following gain values:

$$GPIT = GYAW = .000243 .$$

3.4.3 Control Parameter Improvement

The values of the control parameters γ and β were altered (using the Hooke - Jeeves Algorithm) to improve performance in the 6DOF simulation. Rather than attempt to incorporate the 6DOF simulation as a function subprogram to evaluate $F(\gamma,\beta)$, it was decided to bypass the anticipated programming difficulty by just running the simulation any time γ or β was changed and $F(\gamma,\beta)$ needed to be re-evaluated for the H-J Algorithm. Although the iterations thus performed were slow, overall it seemed to be the most straightforward way to proceed.

3.5 Sensitivity Analysis

3.5.1 Sensitivity for the Four-State Simulation

When the 3-state control law was used on the 6DOF simulation, a small change in lowering the missile velocity from 1095 ft/sec to 1090 ft/sec resulted in the miss distance jumping to 264'. Clearly, the question of sensitivity had to be investigated--first with the simpler four-state simulation and then with the 6DOF simulation.

It was noticed that the initial conditions for the terminal guidance phase given by Equation (3.19) had been poorly chosen in the sense that the target was out of the field of view, and so an alternate set was used:

$$H_t = 1000' , Y_d = 5000' , \theta = 89^o . \tag{3.29}$$

The velocity was left the same at 1095 ft/sec and the initial attitude angle was increased to 89^o since the missile would be flying approximately horizontal after the initial pre-programmed pitch maneuver.

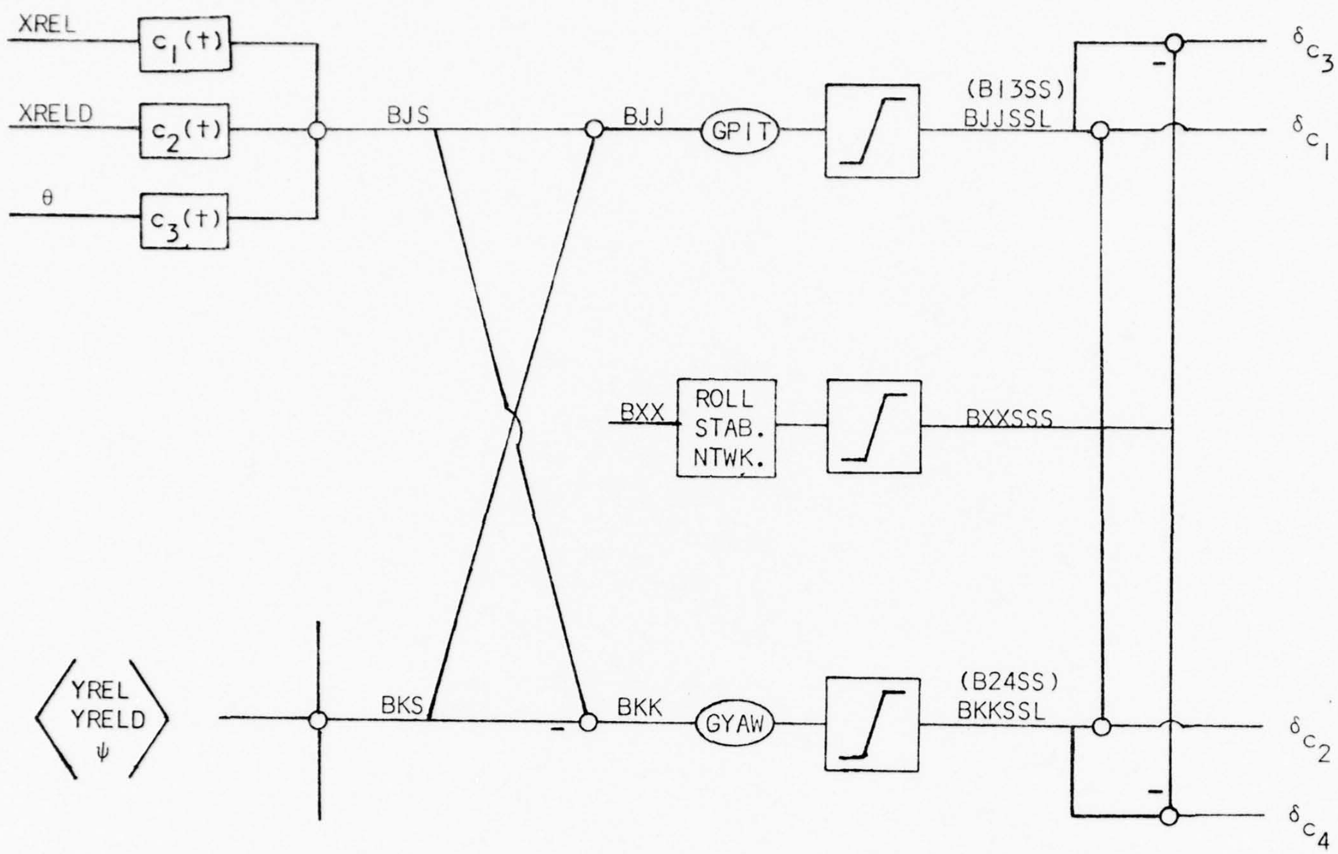

Figure 3.5 Initial Implementation of the 3-State Controller

Varying each of the initial conditions Y_d, H_t, θ, and $V_M(t_o)$ separately, Tables 3.3a, 3.3b, 3.3c, 3.3d were generated using the four-state simulation. As can be seen from them, decreases in the acquisition range are tolerable, but when it is increased to 6500' the simulation becomes unstable. The height can be increased without detriment, but when lowered to 800' instability occurs. The initial attitude angle can only be decreased by two degrees. Performance is most sensitive to decreases in velocity--a decrease of 5 ft/sec of velocity from 1095 ft/sec to 1090 ft/sec causes instability. Recall that the performance obtained with the 6DOF simulation considerably worsened as well when the velocity dropped 5 ft/sec.

The above information suggests that a worst case scenario about which to optimize the control parameters might be:

$$H_t = 1000' \ , \ Y_d = 5000' \ , \ \theta = 87.5^o \ , \ V_M = 1080 \text{ ft/sec} \ . \quad (3.30)$$

An initial attitude angle of 87.5 would hopefully allow a larger window for acceptable initial attitude angles. A lower velocity was chosen as it would be more demanding on the control system. Using the Hooke - Jeeves Algorithm, the optimal control parameters turned out to be

$$\gamma = 5460.992 \quad , \quad \beta = .189291\text{E-}06 \quad (3.31)$$

with a resulting performance of

$$Y_d(t_f) = .915' \quad , \quad \theta(t_f) = 45.012^o \ . \quad (3.32)$$

The four-state simulation was ran in double-precision to minimize the effect of roundoff error on the results. Varying initial conditions one at a time, Tables 3.4a, 3.4b, 3.4c, 3.4d were produced.

The changes in performance when acquisition range, height, or initial attitude angle are altered agree qualitatively with the previous results. Y_d can be decreased and H_t can be increased with performance remaining acceptable. The interval of acceptable initial attitude angles would be from about 86^o to 89.5^o. The one very curious complete reversal of sensitivity is in the variable Y_d. Previously, when Y_d lessened, performance worsened; now, when Y_d increases, performance worsens. With these control parameters, the four-state simulation predicts that the missile should not acquire the target until its velocity decreases below 1080 ft/sec.

3.5.2 Sensitivity with the 6DOF Simulation

The choice of GPIT, GYAW for the 6DOF simulation implementation turned out to be a very delicate one for the new set of initial conditions. Two completely different types of trajectories were observed for various values of GPIT, depending on whether it was above or below some value close to .0004975. When GPIT was less, the missile would overfly the target and the simulation would terminate when the maximum flight time parameter was exceeded. For GPIT greater than this number,

Table 3.3. Performance for Various Initial Conditions (Baseline I)

Table	I.C. Variable	I.C. Value	$Y_d(t_f)$ (ft)	$\theta(t_f)-45$ (deg)	Baseline I (denoted by '*')
3.3a	$Y_d(t_o)$ (ft)	3000.	1.61	-5.31	$H_t(t_o)$ = 1000'
		3500.	4.63	-0.24	
		4000.	2.83	-5.03	$\theta(t_o)$ = 89^o
		4500.	3.09	-0.48	
		5000.*	2.80	-0.00	$Y_d(t_o)$ = 5000'
		5500.	-17.85	-127.60	
		6000.	4.83	2.39	$V_M(t_o)$ = 1095 ft/sec
		6500.	- Unstable -		
3.3b	$H_t(t_o)$ (ft)	800.	- Unstable -		γ = 5535.43
		900.	3.05	-3.35	
		1000.*	2.80	-0.00	β = .18939E-6
		1100.	5.13	1.04	
		1200.	2.29	-6.67	
		1300.	4.77	0.35	
		1400.	4.37	0.58	
		1500.	4.97	4.12	
		1600.	2.79	11.78	
3.3c	$\theta(t_o)$ (deg)	85.	- Unstable -		
		86.	5.95	58.70	
		87.	1.49	-2.99	
		88.	4.07	-3.48	
		89.*	2.80	-0.00	
		90.	2.47	-2.64	
3.3d	$V_M(t_o)$ (ft/sec)	1090.	- Unstable -		
		1095.*	2.80	-0.00	
		1100.	4.14	0.32	
		1105.	5.65	3.79	

Table 3.4. Performance for Various Initial Conditions (Baseline II)

Table	I.C. Variable	I.C. Value	$Y_d(t_f)$ (ft)	$\theta(t_f)-45$ (deg)	Baseline II (denoted by '*')
3.4a	$Y_d(t_o)$ (ft)	3000.	2.41	-4.24	$H_t(t_o)$ = 1000'
		3500.	4.13	-0.07	
		4000.	4.96	0.69	$\theta(t_o)$ = 87.5^o
		4500.	12.98	-6.57	
		5000.*	0.92	0.01	$Y_d(t_o)$ = 5000'
		5500.	16.24	-0.515	
		6000.	- Unstable -		$V_M(t_o)$ = 1080 ft/sec
3.4b	$H_t(t_o)$ (ft)	700.	- Unstable -		γ = 5460.99
		800.	17.45	-33.40	
		900.	-124.35	10.97	β = .18929E-6
		1000.*	0.92	0.01	
		1100.	4.33	0.78	
		1200.	5.57	4.18	
		1300.	3.84	0.81	
		1400.	3.30	-3.86	
		1500.	3.9	-0.08	
		1600.	1.72	-5.12	
3.4c	$\theta(t_o)$ (deg)	85.5	- Unstable -		
		86.5	5.22	0.54	
		87.5*	0.92	0.01	
		88.5	1.23	9.79	
		89.5	5.49	-0.17	
		90.5	- Unstable -		
3.4d	$V_M(t_o)$ (ft/sec)	1065.	4.82	-0.05	
		1070.	5.84	0.87	
		1075.	-0.46	-2.05	
		1080.*	0.92	0.01	
		1085.	- Unstable -		

the missile would impact considerably short of the target. Given the seven digit accuracy of the computer, it was not possible to obtain reasonable performance. See Table 4.1 for a closer examination of this behavior. (The control parameters γ and β were set at 5460.99 and .189291E-6, respectively.)

Table 3.5. Performance Dependence on GPIT

GPIT	RXE(ft)	RZE(ft)	$\theta(t_f)$(deg)
.0005	-1210.	1.1	-28.6
.000499	-1144.	.8	-27.5
.000498	-945.	.4	-46.1
.000497	530.	-567.	$t \geq t_{max}$
.000495	529.	-591.	$t \geq t_{max}$

The control parameters were then altered in the hope that performance could be improved and sensitivity reduced. With

$$\text{GPIT} = .497843 \tag{3.33}$$

and the missile impacting 510.6' short of the target, several runs were made with various values of the control parameters γ,β. As can be seen from Table 4.2, small changes can result in a totally different trajectory.

Table 3.6. Performance Dependence on γ and β

γ	β	RXE(ft)	RZE(ft)	$\theta(t_f)$(deg)
5460.996	.1895E-6	-510.6	.5	-62
5461.	.1895E-6	536.	-380.	$t \geq t_{max}$
5460.996	.1910E-6	-118.	178.	-27.9
5460.996	.1920E-6	536.	-408.	$t \geq t_{max}$

Despite the small miss distance and excellent impact angle achieved with the four-state simulation, it proved to be impossible to achieve reasonable performance through an appropriate choice of the gain placed in the pitch and yaw channels. Since varying the control parameters was unsuccessful as well, it was decided to return to the alternate set of initial conditions for which the missile was 14' above ground with an attitude angle of 38.1° below the horizon.

Initially, the control parameters were

$$\gamma = 5525.508 \ , \ \beta = .19E-6 \ , \ GPIT = .243E-3 \ . \qquad (3.34)$$

Using the Hooke - Jeeves Algorithm, the miss distance was reduced to 10.4' with an attitude angle of -38.2^{o} when the control parameters were 5525.45 and .160E-6, respectively.

Different runs were made on the 6DOF simulation with various initial conditions. Table 3.7 contains the results. The first line represents the baseline case.

Table 3.7. 6DOF Simulation Runs for Various Initial Conditions

$Y_d(t_o)$	$H_t(t_o)$	$\theta(t_o)$	$V_M(t_o)$	RXE	RZE	$\theta(t_f)$
-5000.	-1500.	-5.	1090.856	.3	-10.4	-38.1
~~-4500.~~	~~-1500.~~	~~-5.~~	~~1090.856~~	~~1.6~~	~~-68.9~~	~~-44.~~
-5500.	-1500.	-5.	1090.856	-1065.3	1.2	-77.5
-5000.	-1400.	-5.	1090.856	-690.7	.5	-47.2
-5000.	-1600.	-5.	1090.856	-116.7	1.5	-89.7
-5000.	-1500.	-4.	1090.856	.7	-150.4	-32.4
-5000.	-1500.	-6.	1090.856	-495.1	.9	-41.8
-5000.	-1500.	-5.	1085.856	.9	-127.1	-35.5
-5000.	-1500.	-5.	1095.856	-448.7	1.2	-43.4

As can be seen from Table 3.7, performance is very sensitive to any change in the initial condition. Such sensitivity was not observed with the four-state simulation. This is probably due to the fact that in the simpler simulation there was no hard constraint on the controller u; whereas, in the 6DOF simulation a bound of $\pm 10^{o}$ existed for tail fin displacement.

Figures 3.6 and 3.7 represent the baseline trajectory.

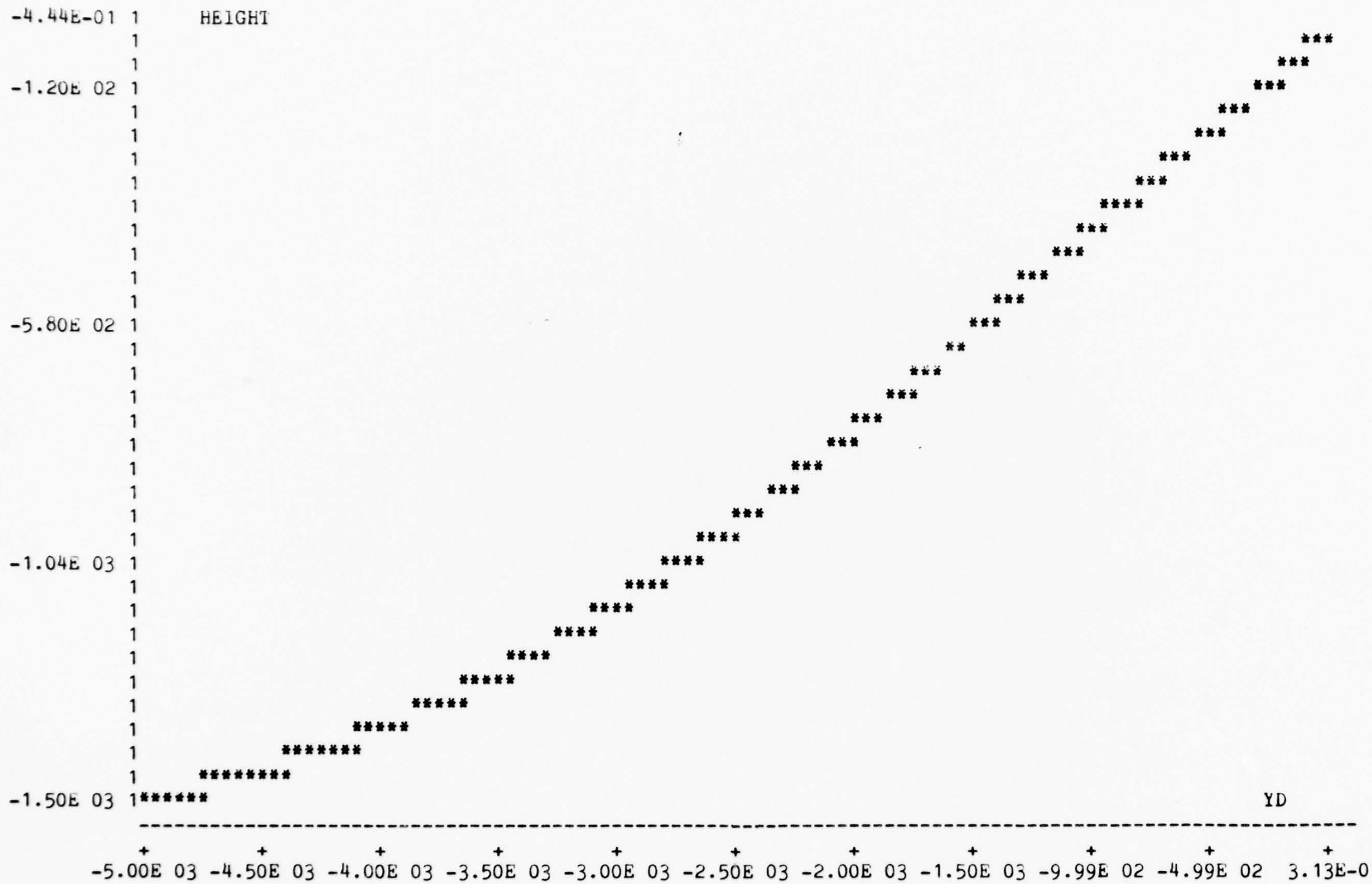

Figure 3.6 RZE vs. RXE

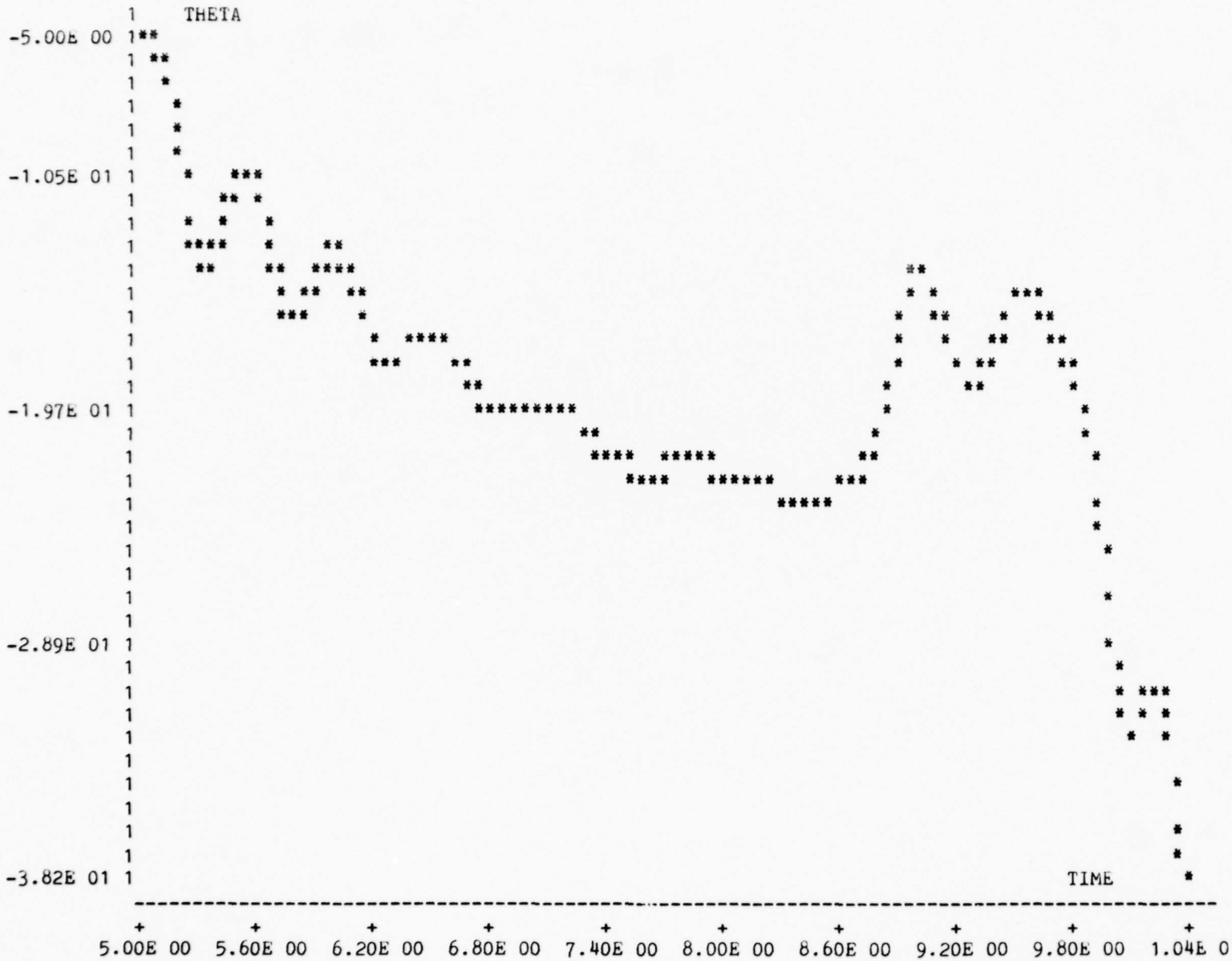

Figure 3.7. THETA vs. TIME

Chapter IV

CONTROL LAW IMPLEMENTATION - PRACTICAL CASE

4.1 Disadvantages of the Three-State Controller

Only one of the three states used to control the missile can be measured, the attitude angle θ. Both range Y_d and range rate $\dot{Y}_d$ are not available as there is no radar or other such measuring devices present. The laser guided missile can detect the reflected laser beam and would have $\dot{\lambda}$ (the time derivative of the line-of-sight angle λ) approximated by the seeker. This alternate set of state variables $(\lambda, \dot{\lambda}, \theta)$ would seem to be more attractive than the set $(Y_d, \dot{Y}_d, \theta)$ because of the possibility of using actual physical measurements.

4.2 A Controller Using λ, $\dot{\lambda}$

Figure 4.1 gives the relationship between the various state variables.

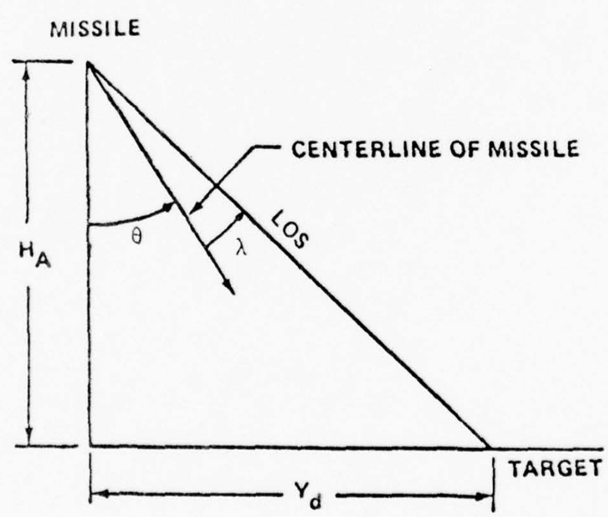

Figure 4.1. Missile - Target Geometry

Using Figure 4.1, the following equation is obtained.

$$Y_d = H_t \tan(\theta+\lambda) \ . \tag{4.1}$$

Differentiating,

$$\dot{Y}_d = H_t \sec^2(\theta+\lambda)(\dot{\theta}+\dot{\lambda}) + \tan(\theta+\lambda)\dot{H}_t \ . \tag{4.2}$$

Substituting, the control law becomes of the form

$$u = f(\lambda, \dot{\lambda}, \theta, \dot{\theta}, H_t, \dot{H}_t) \tag{4.3}$$

where

$$f(\lambda, \dot{\lambda}, \theta, \dot{\theta}, H_t, \dot{H}_t) = c_1 H_t \tan(\theta+\lambda) + c_2 (H_t \sec^2(\theta+\lambda)(\dot{\theta}+\dot{\lambda}) + \tan(\theta+\lambda)\dot{H}_t) + c_3\theta \ . \tag{4.4}$$

4.3 Implementation of the λ, $\dot{\lambda}$, θ Controller

Initially, all states were assumed to be known. Under this assumption, the following performance was obtained:

$$Y_d(t_f) = **** \quad , \qquad \theta(t_f) = **** \quad . \tag{4.5}$$

As for implementing the two pairs of variables ($\lambda, \dot{\lambda}$ and $\theta, \dot{\theta}$), one of each pair will be measured physically, and the other member will be approximated by some appropriate signal processing--either an integration or differentiation network.

As for the last pair, neither H_t nor $\dot{H}_t$ can be measured directly. It is possible, however, to approximate H_t, if the attitude and velocity of the missile are known, by using

$$\dot{H}_t = -V_M \cdot \cos\theta \ . \tag{4.6}$$

If the initial height is known, Equation (4.6) could be used to update the estimate of H_t. The only way that this might be done would be in the design of the preprogrammed pitch maneuver.

There seems to be no way to avoid the need for range and range rate information. Equation (4.6) seems to come as close as possible to drawing upon variables that can be measured. The velocity V_M is not measured but does behave nicely throughout the flight, almost a linear function of t. Perhaps a profile of V_M could be stored for or generated during the flight.

Note: The "****" appearing in Equation (4.5) have been inserted since the code had some errors in it. These numbers should be close to those previously obtained since the two formulations are mathematically equivalent. This approach was not followed up due to the sensitivity problem discussed in Chapter III. It was felt that it was important to outline the approach to be used.

4.4 Estimation of Time-to-go

The problem of estimating time-to-go that is needed for the control law remains even if all other problems are resolved. A fixed estimate could be used, but it would have to vary depending on the initial conditions.

A second approach might be to use H_t and $\dot{H}_t$, but since these two variables are being estimated, this method probably would not fare too well.

A third alternate, perhaps, is to use the seeker. If the intensity of the reflected laser beam can be measured, along with the rate at which the intensity is varying, then an estimate of time-to-go can be obtained.

Intensity I is inversely proportional to the square of the distance from the missile to the target, the slant range R. Thus

$$I = k/R^2 \ . \tag{4.7}$$

Differentiating with respect to time yields,

$$dI/dt = -2kR^{-3}dR/dt \ . \tag{4.8}$$

Hence,

$$\text{time-to-go} = \frac{R}{dR/dt} = -\frac{2I}{dI/dt} \ . \tag{4.9}$$

Chapter V

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDY

Even assuming perfect knowledge of the states Y_d, $\dot{Y}_d$, θ, the best performance that could be obtained with the 6DOF simulation was

$$RZE = -10.4 \quad , \quad \theta(t_f) = -38.1 \ . \tag{5.1}$$

When one considers that the center of mass of the target is about 5' above ground, the performance is very close to satisfying the criteria for success. When actual measurements are used, one can only expect performance to deteriorate.

It must be kept in mind that the above was assuming that all states could be measured exactly. When actual measurements are used for some of the variables and estimates for others, the performance will degrade. If the three-state controller is to work, these errors should be quite small.

The other observation that needs to be made is that the performance is extremely sensitive to changes in the initial conditions when the target is acquired. Even if the miss distance and attitude angle at impact were zero, this sensitivity would defeat any practical implementation of the control law. Small changes in the control parameters also resulted in large changes in performance; that is, the assumption of continuity that was made in determining the control parameters is not valid for the actual system as represented by the 6DOF simulation.

The only conclusion that can be reached is that the three-state controller cannot be satisfactorily implemented in the 6DOF simulation. The main explanation the authors feel is that there is a hard constraint on the controller in the 6DOF simulation, but not in the four-state simulation. This, more than the assumptions of neglecting the angle of attack, or no lag in the autopilot, dictates that the implementation will not prove to be satisfactory. The control parameters can be adjusted so that a successful flight is achieved for most sets of initial conditions, but the extreme sensitivity makes this implementation of no value.

The authors feel that it would be worthwhile to reformulate and solve the control problem so that a hard constraint is present on the controller. Such a controller (which will be of the bang-bang variety) should be implemented, assuming again that all states can be measured, to verify whether or not the presence of the hard constraint on the tail fin deflection is the main culprit for the sensitivity problem. If successfully implemented, then the problem of practical implementation of such a control law could be examined.

Chapter VI

SIMPLIFIED 6DOF SIMULATION

This listing is complete except for the omission of subroutines COSD, SIND, ATAND which are the same as the FORTRAN library subroutines but work with degrees instead of radians.

The old subroutine DUMMY has been relabeled TIMEV. 'TIMEV' was placed as an entry point in DUMMY but the presence of arguments caused a syntax error which was most easily corrected by renaming subroutine DUMMY.

Since the altered simulation does not have the random error disturbance capability, the main program could be shortened by omitting the second half which is the statistical analysis part for Monte Carlo sets.

```
C
C*****DIMODS TO BE USED WITH FORTRAN AMRK INTEGRATION ROUTINE
C
      COMMON C(3830),GRAPH(300,4)
      REAL KA,K1
      COMMON/CEPASS/X(100),Y(100)
      EQUIVALENCE (C(2662),HMIN  ), (C(2663),HMAX  ), (C(2664),DER(1)),
     C            (C(2561),N     ), (C(2562),IPL(1)), (C(2965),VAR(1)),
     C            (C(2000),T     ), (C(2011),KSTEP ), (C(2010),STEP  ),
     C            (C(2012),LSTEP ), (C(2008),PLOTNO), (C(2009),NOPLOT),
     C    (C(2023),OPOINT), (C(2025),TIME(1)), (C(2325),VLABLE(1,1)),
     C            (C(3167),NOOUT ), (C(2022),OPTN10), (C(2006),REPPLT),
     C            (C(2865),EU(1)), (C(2765),EL(1)), (C(2007),PTLESS)
      EQUIVALENCE                  (C(1971),RITE  ), (C(1972),RKUTTA)
      EQUIVALENCE (C(1973),KASE  ), (C(1974),NJ    ), (C(1975),NPT   )
      EQUIVALENCE (C(3512),ISGCT),(C(3721),ITCT),(C(3511),RNSTRT)
      EQUIVALENCE (C( 21),IBVNSW)
      EQUIVALENCE (C( 22), IPLOT)
      EQUIVALENCE (C( 23),XLAMBD)
      EQUIVALENCE (C( 24), KSSIG)
      EQUIVALENCE (C( 25),CEPSIG(1))
      EQUIVALENCE (C(300),RMISS)
      DIMENSION RMISST(100)
      EQUIVALENCE (C(1000),RMISST(1))
      EQUIVALENCE (C(301),      L)
      EQUIVALENCE (C(302),  RYF)
      EQUIVALENCE(C(19),PSIZE)
      EQUIVALENCE (C(303),  RZF)
      EQUIVALENCE (C( 31),LCEP)
      EQUIVALENCE (C(3825),  NCASE), (C( 625),    IBL)
      EQUIVALENCE (C(2662)  ,DERSV)
      EQUIVALENCE (C(3000),VSD(1))
      EQUIVALENCE (C(3010),VMEAN(1))
      EQUIVALENCE (C(3020),IMVNDX(1))
      EQUIVALENCE (C(3030),IMVCT )
      REAL KSSIG
      INTEGER CEPSIG
      DIMENSION CEPSIG(6)
      DIMENSION TIME(300)
      DIMENSION   VLABLE(2,15)    , IPL(100)        , DER(101)
      DIMENSION   VAR(101)        , EL(100)         , EU(100)
      DIMENSION IMVNDX(10), VMEAN(10), VSD(10)
      EQUIVALENCE (C(1980),RN    )
      EQUIVALENCE (C(1981),RNT   )
      EQUIVALENCE (C(1982),PLOTN4)
      EQUIVALENCE (C(1983),PLOTN2)
      EQUIVALENCE (C(1984),NPLOT )
C
      EQUIVALENCE (C(2020),LCONV)
      EQUIVALENCE (C(1651),RTXE),(C(1655),RTYE),(C(1659),RTZE),
     .(C(1615),RXE),(C(1619),RYE),(C(1623),RZE)
      INTEGER OPOINT
      INTEGER OPT
      EXTERNAL AUXSUB
C  ***************************WKU CHANGE**************
      DO 1111 I = 1,3830
 1111 C(I) = 0.0
C ****************************WKU CHANGE**************
      ISGCT=0
```

```
      ISGCT=1
      ITCT=0
      ITCT=1
      MODE=-1
      ITSNDX=0
C
C
C                     THIS CALL TO SUBROUTINE RANNUM IS TO PERMIT USE OF
C                     DIFFERENT RANDOM NUMBER GENERATOR STARTERS (IRNST).
C                     IF SUBROUTINE NORMAL IS CALLED WITHOUT FIRST CALLING
C                     RANNUM, THE RANDOM NUMBER SEQUENCE WILL ALWAYS
C                     BE STARTED WITH THE SAME NUMBER (ENTERED AS A DATA
C                     STATEMENT IN SUBROUTINE NORMAL), WHICH WILL RESULT
C                     IN THE SAME SEQUENCE ALWAYS BEING GENERATED.
      L = 1
      NP = 0
      CALL COUNTV
 1000 CALL ZERO
 1001 IF(PLOTNO.LE.0.)GOTO7
      IF(REPPLT.GT.0.)GOTO7
C
C                               REPPLT = 0.  USE NEW NO.4,7 (DISCARD OLD)
C                                        1.  USE OLD PLUS THOSE ADDED
C                                       -1.  USE NEW NO. 7 (DISCARD OLD)
      IF (REPPLT.GT.-1.0) NOOUT = 0
      NPLOT=0
    7 CALL OINPT1
8     CONTINUE
      CALL RANNUM(1.,RNSTRT,DUM)
      IF(ISGCT.GT.0) CALL MCARLO (DUM, -1, IDUM)
      C(1976) = 1.
      KASE=0
      LSTEP = STEP
      NPLOT4=PLOTN4
      NPLOT2=PLOTN2
      NOPLOT=PLOTNO
      NOPLOT=NPLOT
 1002 CALL  SUBL1
 1003 CALL  AUXI
 1004 CALL  SUBL2
 1005 CONTINUE
      DER(101)=DER(1)
      IF(DER(1).GT.DERSV)DER(1)=DERSV
      C(1976)=1.
 1006 CALL   AUXSUB
 1007 NJ=N-1
      CALL AMRK(AUXSUB)
 1008 CONTINUE
 1009 CONTINUE
      RDELX=RTXE-RXE
      RDELY=RTYE-RYE
      RDELZ=RTZE-RZE
      IF(RDELZ .LT. 0. .OR. RDELX .LT. 0.) LCONV=2
      CALL SUBL3
      IF ( KSTEP .EQ. 1 ) GO TO 1007
C
C
C
C
C
```

```
C
C*****************SAVE MISS DISTANCE FOR EACH RUN OF THE MONTE CARLO
C                 RUN SET*****************************************************
C
C
C
      IF(LCEP.NE.1) GO TO 20
      LCEP=0
      NP = NP + 1
      X(NP) = RYF
      Y(NP) = RZF
      RMISST(NP) = RMISS
      GO TO 21
   20 CONTINUE
      WRITE(6,805)
  805 FORMAT(1H0,///)
      WRITE(6,22) L
   22 FORMAT(1H0,11X,26H*$*$*$ WARNING RUN NUMBER ,I3,
     2       37H DID NOT INTERSECT TARGET PLANE*$*$*$,/,
     3     //,26X,38H$*$*THIS RUN DROPPED FROM DATA SET*$*$)
      WRITE(6,805)
   21 CONTINUE
      L = L + 1
C
C
C
C^^^^^^CONTROL PARAMETER OUTPUT
      PRINT 466,C(2916),C(2917),C(2918),C(2919),C(2920),C(2921)
  466 FORMAT(1H0,'  GAMMA = ',E15.6,4X,'  BETA= ',E15.6,/,
     1'   KA = ',F10.6,4X,'  K1 = ',F10.6,4X,'  W1 = ',F10.6,/,
     2'   GPIT = ',F10.6)
      THOFF = C(350)+45
      WRITE(6,1212) C(350), THOFF
 1212 FORMAT('OTHETA MISSILE IN DEGREES =',E15.7
     .,/,' VARIATION FROM DESIRED 45 DEGREE IMPACT=',E15.7,//)
C
C   *********WKU PLOTTING ROUTINE*********
C
      CALL WKPLOT
C
C   *********END WKU PLOTTING ROUTINE*****
C
C
C
C
      CALL PROCES
      IF (OPTN10 .GT. 0.) CALL DUMPO
      CALL RESET
      IF(LSTEP.EQ.5.OR.LSTEP.EQ.7.OR.NOPLOT.EQ.0)GOTO5
      CALL TIMEV(DELT)
      WRITE(6,96)DELT
   96 FORMAT(1H ,17HSTART PLOTTING AT,F14.7)
      LESSPT=PTLESS
      OPOINT=OPOINT-LESSPT
C
      CALL PLOT4(GRAPH,OPOINT,VLABLE,TIME,NPLOT4,NPLOT2,NOPLOT)
      CALL PLOT2
      CALL PLOTN
      CALL TIMEV(DELT)
      WRITE(6,97)DELT
```

```
   97 FORMAT(1H ,18HPLOTTING ENDED AT ,F14.7)
    5 GO TO (1000,1001,1002,1003,1004,1005,1006,1007,1008,1009,1010),
     1 LSTEP
 1010 CONTINUE
C
C
C**** MEAN VARIANCE AND STANDARD DEVIATION
C
C
C
       WRITE(6,100)
  100 FORMAT(1H1,13X,36HMEAN VARIANCE AND STANDARD DEVIATION/,
     1 7X,10HC-LOCATION,9X,8HMEAN VAR,19X,7HSTD DEV/)
      DO 120 I=1,IMVCT
      ILOC = IMVNDX(I)
      WRITE(6,102) ILOC,VMEAN(I),VSD(I)
  102 FORMAT(10X,I5,8X,E15.8,11X,E15.8)
  120 CONTINUE
C
C
C
C
C
C************MONTE CARLO AND CEP LOGIC FOLLOWS*************************
C
C
C
      IF(NP.LT.NCASE.AND.L.LE.(NCASE+5))WRITE(6,807)
      IF(NP.LT.NCASE.AND.L.LE.(NCASE+5))GO TO 8
  807 FORMAT(1H1,3(/),39H          THIS RUN ADDED DUE TO BREAKLOCK,3(/))
      J =0
      WRITE(6,800)
  800 FORMAT(1H1, 96X,10HY-MISS     ,10HZ-MISS     ,10HMISS DIST /,
     1     6X,123H-------------------------------------------------
     2------------------------m--------------------------------------------
     3--)
      DO 801 I=1,NP
      J = J +1
      WRITE(6,802) X(I),Y(I),RMISST(I)
  802 FORMAT( 6X,10H1          ,10H1          ,10H1          ,10H1
     1,10H1          ,10H1          ,10H1          ,10H1          ,10H1
     2     ,1H1,F9.5,2H 1,F9.5,2H 1,F8.5,2H 1)
      WRITE(6,803)
  803 FORMAT( 6X,123H-------------------------------------------------
     2-------------------------------------------------------------------
     3------)
      IF(J.GT.30) WRITE (6,800)
      IF(J.GT.30) J = 0
  801 CONTINUE
      IF(IBL.LE.0)GO TO 804
      L=L-1
      XIBL=IBL
      XL=L
      RATIO=XIBL/XL
      WRITE(6,806)IBL,L,RATIO
  806 FORMAT(1H1,15(/),1X,10(11H*BREAKLOCK*),
     .        /,1X,11H*BREAKLOCK*,3X,16HTHIS RUN SET HAD,I4,25HBREAKLOCK
     .FLIGHTS OUT OF ,I4,23HGIVING A PROPORTION OF.,F6.4,4X,11H*BREAKLOC
     .K ,
     .     /,1X, 9HBREAKLOCK ,3X,17HTHIS RUN SET HAD ,I4,24HBREAKLOCK FL
```

```
     .IGHTS OUT OF ,I4  ,23HGIVING A PROPORTION OF ,F6.4,4X,9HBREAKLOCK,
     .        /,1X,11H*BREAKLOCK*,88X,11H*BREAKLOCK*,
     .        /,1X,11H*BREAKLOCK*,88X,11H*BREAKLOCK*,
     .        /1X,10(11H*BREAKLOCK*))
804   CONTINUE
      CALL CEPAS(NP,IBVNSW,IPLOT,XLAMBD,KSSIG,CEPSIG,PSIZE)
 4668 CONTINUE
      CALL EXIT
      STOP
      END
```

```
      SUBROUTINE AMRK(AUXSUB)
      COMMON C(3830)
      DIMENSION CSAV(100), IPL(100)
      REAL K1(100), K2(100), K3(100), K4(100)
      EQUIVALENCE (C(2000),     T)
      EQUIVALENCE (C(2664), DELT)
      EQUIVALENCE (C(1974),    NJ)
      EQUIVALENCE (C(2562),  IPL(1))
      EQUIVALENCE (C(1976),XNDRK)
      XNDRK = -1.
      DO 1 I = 1,NJ
      J = IPL(I)
C
C****STORE INITIAL VALUES
      CSAV(I) = C(J+3)
C
C*** COMPUTE K1
      K1(I) = DELT*C(J)
    1 C(J+3) = CSAV(I) + .5*K1(I)
      T = T + .5*DELT
      CALL AUXSUB
C
C*** COMPUTE K2
      DO 2 I = 1,NJ
      J = IPL(I)
      K2(I) = DELT*C(J)
    2 C(J+3) = CSAV(I) + .5*K2(I)
      CALL AUXSUB
C
C*** COMPUTE K3
      DO 3 I = 1,NJ
      J = IPL(I)
      K3(I) = DELT*C(J)
    3 C(J+3) = CSAV(I) + K3(I)
      T = T + .5*DELT
      CALL AUXSUB
C
C*** COMPUTE K4
      DO 4 I = 1, NJ
      J = IPL(I)
      K4(I) = DELT*C(J)
    4 C(J+3) = CSAV(I) +(K1(I) + 2.*(K2(I) + K3(I)) + K4(I))/6.
      XNDRK = 1.
      CALL AUXSUB
      RETURN
      END
```

```
      SUBROUTINE AUXI
      COMMON C(3830)
      EQUIVALENCE (C(2361),NOMOD ),(C(2362),XMODNO(1)),(C(2561),N)
      DIMENSION  XMODNO(99)
      N = 1
      DO 1 I=1,NOMOD
      L =XMODNO(I)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
     1      ,24,25,26,27,28,29,30,31,32,33,34,35,36,37),L
    2 CALL A1I
      GO TO 1
    3 CALL A2I
      GO TO 1
    4 CALL A3I
      GO TO 1
    5 CALL A4I
      GO TO 1
    6 CALL A5I
      GO TO 1
    7 CALL C1I
      GO TO 1
    8 CALL C2I
      GO TO 1
    9 CALL C3I
      GO TO 1
   10 CALL C4I
      GO TO 1
   11 CALL C5I
      GO TO 1
   12 CALL C6I
      GO TO 1
   13 CALL C7I
      GO TO 1
   14 CALL C8I
      GO TO 1
   15 CALL C9I
      GO TO 1
   16 CALL C10I
      GO TO 1
   17 CALL D1I
      GO TO 1
   18 CALL D2I
      GO TO 1
   19 CALL D3I
      GO TO 1
   20 CALL D4I
      GO TO 1
   21 CALL D5I
      GO TO 1
   22 CALL G1I
      GO TO 1
   23 CALL G2I
      GO TO 1
   24 CALL G3I
      GO TO 1
   25 CALL G4I
      GO TO 1
   26 CALL G5I
      GO TO 1
```

```
27 CALL G6I
   GO TO 1
28 CALL S1I
   GO TO 1
29 CALL S2I
   GO TO 1
30 CALL S3I
   GO TO 1
31 CALL S4I
   GO TO 1
32 CALL S5I
   GO TO 1
33 CALL S6I
   GO TO 1
34 CALL S7I
   GO TO 1
35 CALL S8I
   GO TO 1
36 CALL S9I
   GO TO 1
37 CALL S10I
 1 CONTINUE
   RETURN
   END
```

```
      SUBROUTINE AUXSUB
      COMMON C(3830)
      EQUIVALENCE (C(2000),T),(C(2361),NOMOD),(C(2362),XMODNO(1))
      EQUIVALENCE (C(2561),N  ), (C(2562),IPL(1)), (C(2664),DER(1))
      EQUIVALENCE (C(2965),VAR(1) )
      EQUIVALENCE (C(2020),LCONV)
      DIMENSION   DER(101)         , VAR(101)         , IPL(100)
      DIMENSION   XMODNO(99)
      EXTERNAL A1
      EXTERNAL A2
      EXTERNAL A3
      DO 1 I=1,NOMOD
      IF(LCONV.EQ.2)RETURN
      L =XMODNO(I)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
     123,24,25,26,27,28,29,30,31,32,33,34,35,36,37),L
    2 CALL A1
      GO TO 1
    3 CALL A2
      GO TO 1
    4 CALL A3
      GO TO 1
    5 CALL A4
      GO TO 1
    6 CALL A5
      GO TO 1
    7 CALL C1
      GO TO 1
    8 CALL C2
      GO TO 1
    9 CALL C3
      GO TO 1
   10 CALL C4
      GO TO 1
   11 CALL C5
      GO TO 1
   12 CALL C6
      GO TO 1
   13 CALL C7
      GO TO 1
   14 CALL C8
      GO TO 1
   15 CALL C9
      GO TO 1
   16 CALL C10
      GO TO 1
   17 CALL D1
      GO TO 1
   18 CALL D2
      GO TO 1
   19 CALL D3
      GO TO 1
   20 CALL D4
      GO TO 1
   21 CALL D5
      GO TO 1
   22 CALL G1
      GO TO 1
   23 CALL G2
```

```
      GO TO 1
   24 CALL G3
      GO TO 1
   25 CALL G4
      GO TO 1
   26 CALL G5
      GO TO 1
   27 CALL G6
      GO TO 1
   28 CALL S1
      GO TO 1
   29 CALL S2
      GO TO 1
   30 CALL S3
      GO TO 1
   31 CALL S4
      GO TO 1
   32 CALL S5
      GO TO 1
   33 CALL S6
      GO TO 1
   34 CALL S7
      GO TO 1
   35 CALL S8
      GO TO 1
   36 CALL S9
      GO TO 1
   37 CALL S10
    1 CONTINUE
      RETURN
      END
```

```
      BROUTINE A1
      COMMON C(3830)
C
C**TABLE LOOKUP FOR AERO COEF
      COMMON
     */NC1Z/NC1(2)     /NC2Z/NC2(4)     /NC3Z/NC3(4)     /NC5Z/NC5(4)
     */CA1/VM1(15)    /CA2/BA2(6),VM2(6)  /CA3/BA3(7),VM3(5)
     */CA5/BA5(7),VM5(3)
     */CZPFZ/CZPF(35) /CZ2FZ/CZ2F(35)  /CMPFZ/CMPF(35)  /CM2FZ/CM2F(35)
     */CY4FZ/CY4F(36) /CN4FZ/CN4F(36)  /CL4FZ/CL4F(21)  /CL2FZ/CL2F(21)
     */CZDFZ/CZDF(35)                  /CMDFZ/CMDF(35)
     */CMQFZ/CMQF(36)  /CLPFZ/CLPF(36)  /CLDFZ/CLDF(21)
     */CXOFZ/CXOF(15)
      COMMON  /CMOFZ/CMOF(6)  /CA4Z/CA4(6)
      COMMON /NC6Z/NC6(4)  /CA6/BA6(6),VM6(4) /DXCPFZ/DXCPF(24)
      COMMON /NC7Z/NC7(4)  /CA7/BA7(9),VM7(3) /DCMFZ/DCMF(27)
      COMMON /NC8Z/NC8(2)  /CA8Z/CA8(3)  /DCMOFZ/DCMOF(3)
C
C**INPUT DATA
      EQUIVALENCE (C(1200),OPTA1 )
      EQUIVALENCE (C(1240),CL1   )
      EQUIVALENCE (C(1241),CL2   )
      EQUIVALENCE (C(1242),CL3   )
      EQUIVALENCE (C(1243),DCM1F(1))
      EQUIVALENCE (C(1246),CL0   )
      EQUIVALENCE (C(1307),RFLGTH)
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(0204),VMACH )
      EQUIVALENCE (C(0367),BALPHA)
      EQUIVALENCE (C(0368),BALPHY)
      EQUIVALENCE (C(0369),BALPHP)
      EQUIVALENCE (C(0370),BPHIP )
      EQUIVALENCE (C(1196),BDELT1)
      EQUIVALENCE (C(1197),BDELT2)
      EQUIVALENCE (C(1198),BDELT3)
      EQUIVALENCE (C(1199),BDELT4)
      EQUIVALENCE (C(1551),OPTM  )
      EQUIVALENCE (C(1555),UDL1  )
      EQUIVALENCE (C(1556),UDL2  )
      EQUIVALENCE (C(1557),UDL3  )
      EQUIVALENCE (C(1558),UDL4  )
C
C**INPUTS FROM MAIN PROGRAM
      EQUIVALENCE (C(2000),T     )
      EQUIVALENCE (C(2020),LCONV)
C
C**OUTPUT TO MODULES
      EQUIVALENCE (C(1200),OPTHNG)
      EQUIVALENCE (C(1203),CX    )
      EQUIVALENCE (C(1204),CY    )
      EQUIVALENCE (C(1205),CZ    )
      EQUIVALENCE (C(1206),CLP   )
      EQUIVALENCE (C(1207),CMQ   )
      EQUIVALENCE (C(1208),CNR   )
      EQUIVALENCE (C(1209),CL    )
      EQUIVALENCE (C(1210),CM    )
      EQUIVALENCE (C(1211),CN    )
C
```

```
       EQUIVALENCE (C(1237),CCQ)
       EQUIVALENCE (C(1238),CCR)
      EQUIVALENCE (C(1239),CCRTOQ)
      EQUIVALENCE  (C(1257),SDP)
      EQUIVALENCE  (C(1267),SDQ)
      EQUIVALENCE  (C(1277),SDR)
C**OTHER OUTPUTS
      EQUIVALENCE (C(1212),CXO    )
      EQUIVALENCE (C(1213),CZO    )
      EQUIVALENCE (C(1214),DCZ2   )
      EQUIVALENCE (C(1215),CZDQ   )
      EQUIVALENCE (C(1216),CZDR   )
      EQUIVALENCE (C(1217),DCY4   )
      EQUIVALENCE (C(1218),CMO    )
      EQUIVALENCE (C(1219),DCM2   )
      EQUIVALENCE (C(1220),CMDQ   )
      EQUIVALENCE (C(1221),CMDR   )
      EQUIVALENCE (C(1222),DCN4   )
      EQUIVALENCE (C(1223),DCL1   )
      EQUIVALENCE (C(1224),DCL4   )
      EQUIVALENCE (C(1225),CLDP   )
      EQUIVALENCE (C(1226),VM     )
      EQUIVALENCE (C(1227),BAP    )
      EQUIVALENCE (C(1228),BDL    )
      EQUIVALENCE (C(1229),BDM    )
      EQUIVALENCE (C(1230),BDN    )
      EQUIVALENCE (C(1231),BDP    )
      EQUIVALENCE (C(1232),BDQ    )
      EQUIVALENCE (C(1233),BDR    )
      EQUIVALENCE (C(1236),CH1    )
      EQUIVALENCE (C(1237),CH2    )
      EQUIVALENCE (C(1238),CH3    )
      EQUIVALENCE (C(1239),CH4    )
      EQUIVALENCE (C(1240),CH11   )
      EQUIVALENCE (C(1241),CH21   )
      EQUIVALENCE (C(1242),CH31   )
      EQUIVALENCE (C(1243),CH41   )
      DIMENSION NDCLD(2), DCLDVM(3), DCLDOF(3), DCLDAF(3)
      DIMENSION DCLDFF(3), DCM1F(3)
      EQUIVALENCE (C(1234), DCLDO)
      EQUIVALENCE (C(1235), DCLDA)
C****** YAW PARAMETER INPUT
      EQUIVALENCE (C(2901),OPTNYW)
C
      DATA WATLB1/4HCXO /,WATLB2/4HCMO /,WATLB3/4HCZO /,WATLB4
     #/4HCMO /
      DATA WATLB5/4HDCM2/,WATLB6/4HCZDQ/,WATLB7/4HCMDQ/,WATLB8/
     #4HDCZ2/
      DATA WATLB9/4HDCY4/,WATL10/4HDCN4/,WATL11/4HCLP /,WATL12/
     #4HCMQ /
      DATA WATL13/4HCNR /,WATL14/4HCLDP/,WATL15/4HDCL4/
      DATA WATL16/4HDCL2/,WATL17/4H    /,WATL18/4HDXCP/
      DATA WATL19/4HDCM /
      DATA NDCLD /3,0 /
      DATA DCLDVM / .500,.950, 1.250/
      DATA DCLDOF /.000,-.0013,.0010/
      DATA DCLDAF/-.0016,-.0023,-.0010/
      DATA DCLDFF/.06,.10,.04/
      DATA SDPCC,SCCRQ/0.0,0.0/
C
```

```
C  MULTIPLE ANGLE FORMULAE AND ABSOLUTE VALUES OF ANGLE OF ATTACK
      USPHI = SIND(BPHIP)
      UCPHI = COSD(BPHIP)
      US2PHI = SIND(2.*BPHIP)
      US4PHI = SIND(4.*BPHIP)
      US2PH2 = US2PHI**2
C
C**LIMIT TABLE ARGUMENTS
      BDP=(-BDELT1-BDELT2+BDELT3+BDELT4)/4+SDP+SDPCC
      BDQ=(+BDELT1+BDELT2+BDELT3+BDELT4)/4+SDQ+SCCRQ
      BDR=(-BDELT1+BDELT2-BDELT3+BDELT4)/4+SDR
          SDPCC=CCQ*BDQ+CCR*BDR
      SCCRQ=CCRTOQ*BDR
      BAP = BALPHP
      UAL = ABS(BALPHA)
      UBT = ABS(BALPHY)
      VM = VMACH
      IF (BAP.GT.20.)BAP=20.
      IF (UAL.GT.20.)UAL=20.
      IF (UBT.GT.20.)UBT=20.
C
C**TABLE LOOKUP FOR AERO COEF
      IF(T.GT. 0. .AND. C(1976).LT.0.) GO TO 1000
      XF=0.
      CXO=FINTP1( VM,VM1,CXOF,NC1(1),XF,WATLB1)
      XF=0.
      CMO=FINTP1(    VM,CA4,CMOF,NC2(1),XF,WATLB2)
      XF=0.
      CZO=FINTP2(BAP,VM,BA3,VM3,CZPF,NC3(1),NC3(2),NC3(3),XF,WATLB3)
      CMO=FINTP2(BAP,VM,BA3,VM3,CMPF,NC3(1),NC3(2),NC3(3),XF,WATLB4)
      DCM2=FINTP2(BAP,VM,BA3,VM3,CM2F,NC3(1),NC3(2),NC3(3),XF,WATLB5)
      CZDQ=FINTP2(BAP,VM,BA3,VM3,CZDF,NC3(1),NC3(2),NC3(3),XF, WATLB6)
      CZDR = CZDQ
      CMDQ=FINTP2(BAP,VM,BA3,VM3,CMDF,NC3(1),NC3(2),NC3(3),XF,WATLB7)
      CMDR = CMDQ
      DCZ2=FINTP2(BAP,VM,BA3,VM3,CZ2F,NC3(1),NC3(2),NC3(3),XF,WATLB8)
      XF=0.
      DCY4=FINTP2(BAP,VM,BA2,VM2,CY4F,NC2(1),NC2(2),NC2(3),XF,WATLB9)
      DCN4=FINTP2(BAP,VM,BA2,VM2,CN4F,NC2(1),NC2(2),NC2(3),XF,WATL10)
      CLP=FINTP2(BAP,VM,BA2,VM2,CLPF,NC2(1),NC2(2),NC2(3),XF,WATL11)
      XF=0.
      CMQ=FINTP2(UAL,VM,BA2,VM2,CMQF,NC2(1),NC2(2),NC2(3),XF,WATL12)
      XF=0.
      CNR=FINTP2(UBT,VM,BA2,VM2,CMQF,NC2(1),NC2(2),NC2(3),XF,WATL13)
      XF=0.
      CLDP=FINTP2(BAP,VM,BA5,VM5,CLDF,NC5(1),NC5(2),NC5(3),XF,WATL14)
      DCL4=FINTP2(BAP,VM,BA5,VM5,CL4F,NC5(1),NC5(2),NC5(3),XF,WATL15)
      DCL2=FINTP2(BAP,VM,BA5,VM5,CL2F,NC5(1),NC5(2),NC5(3),XF,WATL16)
      XF=0.
      DCLDO=FINTP1(     VM,DCLDVM,DCLDOF,NDCLD(1),XF,WATL17)
      DCLDA=FINTP1(     VM,DCLDVM,DCLDAF,NDCLD(1),XF,WATL17)
      DCLDF=FINTP1(     VM,DCLDVM,DCLDFF,NDCLD(1),XF,WATL17)
      XF=0.
      DXCP=FINTP2(BAP,VM,BA6,VM6,DXCPF,NC6(1),NC6(2),NC6(3),XF,WATL18)
      XF=0.
      DCM=FINTP2(BAP,VM,BA7,VM7,DCMF,NC7(1),NC7(2),NC7(3),XF,WATL19)
      XF=0.
      DCMO=FINTP1(     VM,CA8,DCMOF,NC8(1),XF,WATL17)
      DCM1=FINTP1(     VM,CA8,DCM1F,NC8(1),XF,WATL17)
      CMO = CMO - CMO
```

```
      CMO = CMO + DXCP*CZO*10.8
 1000 CONTINUE
C
C**AERO COEF WIND AXIS
      DCMO=0.0
      CZP = CZO + DCZ2*US2PH2 + CZDQ*UCPHI*BDQ - CZDR*USPHI*BDR
      CMP = CMO + DCM2*US2PH2 + CMDQ*UCPHI*BDQ - CMDR*USPHI*BDR
      CNP =       DCN4*US4PHI + CMDQ*USPHI*BDQ + CMDR*UCPHI*BDR
      CYP =       DCY4*US4PHI + CZDQ*USPHI*BDQ + CZDR*UCPHI*BDR
C
C** TRANSFORMATION FROM WIND TO BODY AXIS
      CX = CXO
      CL = DCL2*US2PHI+ DCL4*US4PHI + CLDP*BDP
      CY =  CYP*UCPHI - CZP*USPHI
      IF(OPTNYW.GT.0.)CY=0.
      CZ = -CYP*USPHI - CZP*UCPHI
      CN =  CNP*UCPHI - CMP*USPHI
      CM = CNP*USPHI + CMP*UCPHI + CMO + DCMO
      DCLDR = DCLDO+ DCLDA*SIN(6.2832*DCLDF*BALPHP)
      IF(OPTA1 .LE. 2.) GO TO 1
      CL = CL + DCLDR*BDR
      RETURN
    1 CL = CL + CLO + (CL1*USPHI+CL2*US2PHI+CL3*SIND(3.*BPHIP))*BAP/8.
      RETURN
      END
```

```
      SUBROUTINE A2
C**AERO FORCE AND MOMENT MODULE   BODY AXES
      COMMON C(3830)
  101 FORMAT(1H0,4X,21HFRONT LUG CLEARS RAIL,5X,3HT =,1PE10.2,5X,
     *       9HREL VEL =,1PE10.2,5X,14HPITCH MOMENT =,1PE10.2)
C
C**INPUT DATA
      EQUIVALENCE (C(1306),RFAREA)
      EQUIVALENCE (C(1307),RFLGTH)
      EQUIVALENCE (C(1316),RLUG  )
      EQUIVALENCE (C(1317),RAIL  )
      EQUIVALENCE (C(1330),AGV   )
      EQUIVALENCE (C(1742),   AMP2),  (C(1746),   AMP1)
      EQUIVALENCE (C(1332),CPHAS )
      EQUIVALENCE (C(1333),AGH   )
      EQUIVALENCE (C(1334),RLGZ  )
      EQUIVALENCE (C(1405),QBURN )
      EQUIVALENCE (C(1627),AGRAV )
C
C**INPUTS FROM OTHER MODULES
      DIMENSION ISNDX(40)
      EQUIVALENCE (C(3634),  ISNDX(1)),  (C(3512),  I3512)
      EQUIVALENCE (C(0203),PDYNMC)
      EQUIVALENCE (C( 204),VMACH )
      EQUIVALENCE (C(0207),VAIRSP)
      EQUIVALENCE (C( 350),BTHT  )
      EQUIVALENCE (C( 380),RANGO )
      EQUIVALENCE (C(1203),CX    )
      EQUIVALENCE (C(1204),CY    )
      EQUIVALENCE (C(1205),CZ    )
      EQUIVALENCE (C(1206),CLP   )
      EQUIVALENCE (C(1207),CMQ   )
      EQUIVALENCE (C(1208),CNR   )
      EQUIVALENCE (C(1209),CL    )
      EQUIVALENCE (C(1210),CM    )
      EQUIVALENCE (C(1211),CN    )
      EQUIVALENCE (C(1236),CH1   )
      EQUIVALENCE (C(1237),CH2   )
      EQUIVALENCE (C(1238),CH3   )
      EQUIVALENCE (C(1239),CH4   )
      EQUIVALENCE (C(1320),FMXTH )
      EQUIVALENCE (C(1321),FMYTH )
      EQUIVALENCE (C(1322),FMZTH )
      EQUIVALENCE (C(1411),FTHX  )
      EQUIVALENCE (C(1412),FTHY  )
      EQUIVALENCE (C(1413),FTHZ  )
      EQUIVALENCE (C(1422),RLCG  )
      EQUIVALENCE (C(1723),CFA23 )
      EQUIVALENCE (C(1735),CFA33 )
      EQUIVALENCE (C(1739),WP    )
      EQUIVALENCE (C(1743),WQ    )
      EQUIVALENCE (C(1737),    FMX),  (C(1741),    FMY), (C(1745), FMZ)
      EQUIVALENCE (C(1747),WR    )
      EQUIVALENCE (C(1748),   FMIX)
      EQUIVALENCE (C(1738),   WPTO)
      EQUIVALENCE (C(1751),   CRAD)
      EQUIVALENCE (C( 626),    VIB)
      EQUIVALENCE (C(2000),T     )
      EQUIVALENCE  (C(1972),RKUTTA)
```

```
      EQUIVALENCE (C(1975),NPT)
C
C**OUTPUTS
      EQUIVALENCE (C(1300),FXBA  )
      EQUIVALENCE (C(1301),FYBA  )
      EQUIVALENCE (C(1302),FZBA  )
      EQUIVALENCE (C(1303),FMXBA )
      EQUIVALENCE (C(1304),FMYBA )
      EQUIVALENCE (C(1305),FMZBA )
      EQUIVALENCE (C(1308),RDELCG)
      EQUIVALENCE (C(1628),DMASS )
C     EQUIVALENCE (C(1748),FMIX  )
      EQUIVALENCE (C(1749),FMIY  )
      EQUIVALENCE (C(1750),FMIZ  )
C
C**OTHER OUTPUTS
      EQUIVALENCE (C(1309),FMH1  )
      EQUIVALENCE (C(1310),FMH2  )
      EQUIVALENCE (C(1311),FMH3  )
      EQUIVALENCE (C(1312),FMH4  )
      EQUIVALENCE (C(1323),FMXLUG)
      EQUIVALENCE (C(1324),FMYLUG)
      EQUIVALENCE (C(1325),FMZLUG)
      EQUIVALENCE (C(3504),OPTN4)
      DATA FLG2,FLG1/0.,0./
C
C**FORCE VECTOR COMPONENTS
      UQS = PDYNMC*RFAREA
      UQSL = UQS*RFLGTH
C
      FXBA=UQS*(-CX)+FTHX
      FYBA=UQS*CY+FTHY
      FZBA=UQS*CZ+FTHZ
C
C** AERO MOMENTS   (NOTE FACTOR OF 2.0 IN DAMPING COEFFICIENT)
      UL2V = 0.
      IF (VAIRSP .GT. 0.) UL2V = RFLGTH/(2.*VAIRSP)
      FMXBA = (CL + CLP*UL2V*WP) * UQSL                + FMXTH
      FMYBA = (CM + CMQ*UL2V*WQ) * UQSL + FZBA*RDELCG + FMYTH
      FMZBA = (CN + CNR*UL2V*WR) * UQSL - FYBA*RDELCG + FMZTH
C
C** CALCULATE HINGE MOMENTS
      FMH1 = CH1*UQSL
      FMH2 = CH2*UQSL
      FMH3 = CH3*UQSL
      FMH4 = CH4*UQSL
C
C**MOMENTS AND FORCES DUE TO LUGS
      IF(QBURN .LE. 0. .AND. RANGO .LE. RAIL+RLUG) GO TO 70
      UFZL2=FZLUG
      FYLUG = 0.
      FZLUG = 0.
      FMXLUG = 0.
      FMYLUG = 0.
      FMZLUG = 0.
      IF (FLG2 .GT.0.) GO TO 74
      FMX=0.
      FMY=0.
      FMZ=0.
      DO 6 I=1,I3512
```

```
      IDO=I
      IF(ISNDX(I).EQ.1743)CALL MCARLO(DUM,1,IDO)
      IF(ISNDX(I).EQ.1747)CALL MCARLO(DUM,1,IDO)
6     CONTINUE
      C(13) = 1.
      WRITE(6,104)WP,WQ,WR
104   FORMAT(1H ,50X,21HTIPOFF RATES--ROLL = ,F6.1,9H PITCH = ,F6.1,
     .  7H YAW = ,F6.1)
      C(2664)=C(2764)
      FLG2 = 1.
      WRITE(6,102) T,VAIRSP,UFZL2
      WRITE(6,103) RANGO
  103 FORMAT(32X,9HRANGO  = ,F6.4)
  102 FORMAT (1H0,36H         REAR LUG CLEARS RAIL    T = ,F8.4,
     *  10HREL VEL = ,F8.3,16H   RAIL FORCE = ,F8.2)
      GO TO 74
   70 IF (RANGO .LE. RAIL) GO TO 72
      RZDD = AGV
      RYDD = AGH
      FYLUG=-(FYBA+DMASS*AGRAV*RYDD-DMASS*(FMZBA*RLCG/FMIZ
     *   -FMXBA*RLGZ/FMIX))/(1. + DMASS*(RLGZ**2/FMIX+RLCG**2/FMIZ))
      FZLUG = -(FZBA + DMASS*AGRAV*(CFA33-RZDD) +  FMYBA*
     *          RLCG*DMASS/FMIY)/(1. + DMASS*RLCG*RLCG/FMIY)
      FMXLUG = FYLUG*RLGZ
      FMYLUG = FZLUG*RLCG
      FMZLUG = FYLUG*RLCG
      IF (FLG1 .GT. 0.) GO TO 74
      FLG1 = 1.
      WRITE(6, 101) T, VAIRSP, FMYLUG
      WRITE(6,103) RANGO
C
C
C
C
C
C                    WPTO--MONTE CARLO VALUE OF TIPOFF ROLL RATE
C                    DTLUG---TIME INCREMENT BETWEEN FIRST LUG AND
C                            LAST LUG DROP OFF TIME(TIPOFF ROLL RATE
C                            OCCURS AT LAST LUG)
C                    FMX---ROLL MOMENT THAT GENERATES  TIPOFF ROLL RATE
      DTLUG=.026
      FMX=WPTO*FMIX/CRAD/DTLUG
5     CONTINUE
C
C
C
C
      GO TO 74
   72 CONTINUE
      RZDD=0.
      RYDD = 0.
      IF(RANGO .LT. RAIL-.3) GO TO 73
      RZDD = AGV
      RYDD = AGH
   73  FYLUG = -(FYBA + DMASS*AGRAV*(CFA23+RYDD))
      FZLUG = -(FZBA + DMASS*AGRAV*(CFA33-RZDD))
      FMXLUG=-FMXBA
      FMYLUG = - FMYBA
      FMZLUG = - FMZBA
      FLG1 = 0.
```

```
      FLG2=0.
   74 CONTINUE
C**TOTAL FORCE AND MOMENTS
      FYBA = FYBA + FYLUG
      FZBA = FZBA + FZLUG
      FMXBA = FMXBA + FMXLUG
      FMYBA = FMYBA + FMYLUG
      FMZBA = FMZBA + FMZLUG
C
C**LAUNCH TRANSIENTS MOMENTS (1-YAW,2-PITCH,3-ROLL MOMENTS)
C
      IF(FLG2.GT.0.)GO TO 75
      IF(VIB.LE.0.)GO TO 75
      CALL LTRAN(T,DELT,AMP2,FMY,WQ0,1,2)
      CALL LTRAN(T,DELT,AMP1,FMZ,WR0,1,1)
75    CONTINUE
      FMXBA=FMXBA+FMX
      FMYBA=FMYBA+FMY
      FMZBA=FMZBA+FMZ
C
      RETURN
      END
```

```
      SUBROUTINE A3
C**ENGINE MODULE
      COMMON C(3830)
C
C
C** INPUT DATA
      EQUIVALENCE (C(1313),RFXCG )
      EQUIVALENCE (C(1314),RFYCG )
      EQUIVALENCE (C(1315),RFZCG )
      EQUIVALENCE (C(1401),BALPHT)
      EQUIVALENCE (C(1402),BPHIT )
      EQUIVALENCE (C(1403),QNALGN)
      EQUIVALENCE (C(1404),PCFTH )
      EQUIVALENCE (C(1405),QBURN )
      EQUIVALENCE (C(1414),CISP  )
      EQUIVALENCE (C(1415),DWT   )
      EQUIVALENCE (C(1416),DWP   )
      EQUIVALENCE (C(1417),RDCGO )
      EQUIVALENCE (C(1418),RDCGF )
      EQUIVALENCE (C(1419),FMIXF )
      EQUIVALENCE (C(1420),FMIYF )
      EQUIVALENCE (C(1421),RLCGO )
      EQUIVALENCE (C(1423),FMIXO)
      EQUIVALENCE (C(1424),FMIYO)
C
C** INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(2000),T     )
C
C** OUTPUTS
      EQUIVALENCE (C(1308),RDELCG)
      EQUIVALENCE (C(1320),FMXTH )
      EQUIVALENCE (C(1321),FMYTH )
      EQUIVALENCE (C(1322),FMZTH )
      EQUIVALENCE (C(1409),UDWP  )
      EQUIVALENCE (C(1410),FTHRST)
      EQUIVALENCE (C(1411),FTHX  )
      EQUIVALENCE (C(1412),FTHY  )
      EQUIVALENCE (C(1413),FTHZ  )
      EQUIVALENCE (C(1422),RLCG  )
      EQUIVALENCE (C(1628),DMASS )
      EQUIVALENCE (C(1748),FMIX  )
      EQUIVALENCE (C(1749),FMIY  )
      EQUIVALENCE (C(1750),FMIZ  )
C
C**STATE VARIABLES AND THEIR DERIVATIVES
      EQUIVALENCE (C(1496),UIMPD )
      EQUIVALENCE (C(1499),UIMP  )
C**LOOK UP TABLE FOR THRUST
      DIMENSION NTH(2), THA(11), THF(11)
      DATA WATLAB/4HHRST/
      DATA NTH/11,0/
      DATA THA/   0.,.025, .125, .250, .750,1.500,1.625,1.750, 2.00, 3.,
     *100./
      DATA THF/.1,1800.,1750.,1650.,1600.,1400.,1250., 600., 300., 0.,
     *0./
C
      IF (QBURN.GT.0.) RETURN
      CALL TABLE(T,THA,THF,NTH(1),XF,WATLAB,FTHRST)
C
```

```
      IF (QNALGN) 20,20,10
   10 USINA=SIND(BALPHT)
      FTHX=FTHRST*COSD(BALPHT)
      FTHY=-FTHRST*USINA*SIND(BPHIT)
      FTHZ=FTHRST*USINA*COSD(BPHIT)
      FMXTH = -FTHY*RFZCG + FTHZ*RFYCG
      FMYTH =  FTHX*RFZCG + FTHZ*RFXCG
      FMZTH = -FTHX*RFYCG - FTHY*RFXCG
      GO TO 30
   20 FTHX=FTHRST
      FTHY=0.
      FTHZ=0.
      FMXTH=0.
      FMYTH=0.
      FMZTH=0.
   30 CONTINUE
C
      UIMPD = FTHRST
      UDWP = UIMP/CISP
C
      DMASS = (DWT+DWP-UDWP)/32.174
      RDELCG = RDCGO - (RDCGO - RDCGF)*UDWP/DWP
C
      FMIX = FMIXO - (FMIXO-FMIXF)*UDWP/DWP
      FMIY = FMIYO - (FMIYO-FMIYF)*UDWP/DWP
      FMIZ = FMIY
      RLCG = RLCGO + RDELCG
      IF (FTHRST .GT. 0.) RETURN
C
      WRITE (6,100) T
  100 FORMAT (//14H BURNOUT TIME=,F8.4,5H SEC.)
      QBURN=1.0
      RETURN
      END
```

```
      SUBROUTINE A3I
      COMMON C(3830)
      DIMENSION IPL(100), ISNDX(40)
      EQUIVALENCE (C(3634), ISNDX(1)), (C(3512), I3512)
      EQUIVALENCE (C( 367),BALPHA)
      EQUIVALENCE (C( 368),BALPHY)
      EQUIVALENCE (C( 370),BPHIP )
      EQUIVALENCE (C(1308),RDELCG)
      EQUIVALENCE (C(1320),FMXTH )
      EQUIVALENCE (C(1321),FMYTH )
      EQUIVALENCE (C(1322),FMZTH )
      EQUIVALENCE (C(1405),QBURN )
      EQUIVALENCE (C(1411),FTHX  )
      EQUIVALENCE (C(1412),FTHY  )
      EQUIVALENCE (C(1413),FTHZ  )
      EQUIVALENCE (C(1415),DWT   )
      EQUIVALENCE (C(1418),RDCGF )
      EQUIVALENCE (C(1419),FMIXF )
      EQUIVALENCE (C(1420),FMIYF )
      EQUIVALENCE (C(1628),DMASS )
      EQUIVALENCE (C(1739),WP    )
      EQUIVALENCE (C(1743),WQ    )
      EQUIVALENCE (C(1747),WR    )
      EQUIVALENCE (C(1748),FMIX  )
      EQUIVALENCE (C(1749),FMIY  )
      EQUIVALENCE (C(1750),FMIZ  )
      EQUIVALENCE (C(2000),     T)
      EQUIVALENCE (C(2561),N     )
      EQUIVALENCE (C(2562),IPL(1))
      EQUIVALENCE (C(1751),   CRAD)
      EQUIVALENCE (C( 626),    VIB)
      EQUIVALENCE (C(1737),    FMX),  (C(1741),    FMY), (C(1745), FMZ)
      DATA IFLG1,IFLG2/0,0/
      IPL(N  ) = 1496
      N = N+1
      C(1499) = 0.
C
      IF (QBURN .GT. 0.) GO TO 10
      CRAD=57.295778
      FMX=0.
      FMY=0.
      FMZ=0.
      WP = 0.
      WQ = 0.
      WR = 0.
      BALPHA = 0.
      BALPHY = 0.
      BPHIP = 0.
C
C MONTECARLO THRUST DIRECTION ERRORS
C
      DO 5 I = 1, I3512
      IDO = I
      IF(ISNDX(I).EQ.1313) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1314) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1315) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1401) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1402) CALL MCARLO (DUM, 1, IDO)
C**MONTE CARLO TIPOFF ROLL,PITCH AND YAW RATES
```

```
      IF(ISNDX(I).EQ.1738)CALL MCARLO(DUM,1,IDO)
      IF(ISNDX(I).EQ.1746)IFLG2=0
      IF(ISNDX(I).EQ.1742)IFLG1=0
5     CONTINUE
C
      IF(VIB.LE.0.)GO TO 6
      CALL LTRAN(T,DELT,C(1746),DUM,WRO,IFLG2,1)
      CALL LTRAN(T,DELT,C(1742),DUM,WQO,IFLG1,2)
      WQ=WQO/FMIYF      *CRAD
      WR=WRO/FMIYF      *CRAD
6     CONTINUE
      IFLG1=1
      IFLG2=1
C
      RETURN
   10 CONTINUE
      FTHRST=0.
      FTHX=0.
      FTHY=0.
      FTHZ=0.
      FMXTH=0.
      FMYTH=0.
      FMZTH=0.
      DMASS = DWT/32.174
      RDELCG = RDCGF
      FMIX = FMIXF
      FMIY = FMIYF
      FMIZ = FMIYF
      RETURN
      END
```

```
      BLOCK DATA
      COMMON
     */NC1Z/NC1(2)      /NC2Z/NC2(4)      /NC3Z/NC3(4)      /NC5Z/NC5(4)
     */CA1/VM1(15)    /CA2/BA2(6),VM2(6)
     */CA3/BA3(7),VM3(5)      /CA5/BA5(7),VM5(3)
      COMMON  /CMOFZ/CMO (6)  /CA4Z/VM4(6)
     */CZPFZ/CZP(35)    /CZ2FZ/CZ2(35)    /CMPFZ/CMP(35)    /CM2FZ/CM2(35)
     */CY4FZ/CY4(36)    /CN4FZ/CN4(36)    /CL4FZ/CL4(21)    /CL2FZ/CL2(21)
     */CZDFZ/CZD1(35)                    /CMDFZ/CMD1(35)
     */CMQFZ/CMQ(36)    /CLPFZ/CLP(36)    /CLDFZ/CLD1(21)
     */CXOFZ/CXO(15)
      COMMON /NC6Z/NC6(4)  /CA6/BA6(6),VM6(4)  /DXCPFZ/DXCPF(24)
      COMMON /NC7Z/NC7(4)  /CA7/BA7(9),VM7(3)  /DCMFZ/DCMF(27)
      COMMON /NC8Z/NC8(2)  /CA8Z/VM8(3)  /DCMOFZ/DCMO(3)
      DATA NC1/15,0/
      DATA NC2/6,6,36,0/
      DATA NC3/7,5,35,0/
      DATA NC5/7,3,21,0/
      DATA VM1/0.0 ,0.4 ,0.6 ,0.7 ,0.8 ,0.85,0.9 ,1.0 ,1.1 ,1.2 ,
     *         1.3 ,1.4 ,1.6 ,1.8 ,2.4 /
      DATA VM2/0.0,0.7,0.9,1.1,1.4,2.0/
      DATA VM3/ .5, .85, .95,1.05,1.25/
      DATA VM4/ .5, .85, .95,1.05,1.25,2./
      DATA VM5/ .5, .95, 1.25/
      DATA BA2/ 0., 4., 8.,12.,16.,20./
      DATA BA3/ 0., 2., 4., 8.,12.,16.,20./
      DATA BA5/ 0., 2., 4., 8., 12.,16.,20./
      DATA CXO/.308,.293,.295,.299,.313,.330,.352,.492,.616,.696,
     *        .752,.791,.850,.859,.832/
      DATA CMO /
     *+0.10 ,+0.25 ,+0.00 ,-0.15 ,-0.05 , .00 /
      DATA CZP/
     * 0.00 , 0.20 , 0.50 , 1.30 , 2.25 , 3.25 , 4.30 ,
     * 0.00 , 0.20 , 0.50 , 1.35 , 2.35 , 3.40 , 4.45 ,
     * 0.00 , 0.25 , 0.60 , 1.45 , 2.45 , 3.55 , 4.70 ,
     * 0.00 , 0.30 , 0.65 , 1.50 , 2.50 , 3.70 , 4.90 ,
     * 0.00 , 0.30 , 0.65 , 1.50 , 2.50 , 3.70 , 5.00 /
      DATA CMP/
     *+0.10 ,+0.40 ,+0.40 ,+0.00 ,-0.65 ,-1.15 ,-1.50 ,
     *+0.25 ,+0.45 ,+0.45 ,+0.10 ,-0.50 ,-0.80 ,-0.85 ,
     *+0.00 ,+0.25 ,+0.20 ,-0.30 ,-0.90 ,-1.00 ,-0.70 ,
     *-0.15 ,+0.00 ,+0.00 ,-0.35 ,-0.90 ,-1.10 ,-0.50 ,
     *-0.05 ,+0.10 ,+0.10 ,-0.20 ,-0.55 ,-0.45 ,-0.50 /
      DATA CZD1/
     * .053 , .049 , .050 , .055 , .056 , .066 , .069 ,
     * .054 , .051 , .050 , .053 , .057 , .062 , .061 ,
     * .052 , .046 , .053 , .055 , .062 , .065 , .066 ,
     * .055 , .053 , .053 , .054 , .056 , .060 , .061 ,
     * .048 , .046 , .045 , .045 , .047 , .050 , .051 /
      DATA CMD1/
     *-.195 ,-.185 ,-.190 ,-.195 ,-.205 ,-.230 ,-.245 ,
     *-.205 ,-.195 ,-.190 ,-.195 ,-.215 ,-.230 ,-.240 ,
     *-.205 ,-.200 ,-.210 ,-.225 ,-.245 ,-.255 ,-.270 ,
     *-.220 ,-.210 ,-.210 ,-.220 ,-.225 ,-.240 ,-.255 ,
     *-.190 ,-.185 ,-.180 ,-.180 ,-.185 ,-.205 ,-.220 /
      DATA CLD1/
     * .0235, .0240, .0255, .0285, .0325, .0320, .0345,
     * .0270, .0280, .0295, .0325, .0355, .0370, .0370,
     * .0285, .0300, .0315, .0335, .0370, .0375, .0345/
```

```
 DATA CY4/
* -0.00, -0.05, -0.14, -0.30, -0.55, -1.05,
* -0.00, -0.06, -0.16, -0.32, -0.57, -1.07,
* -0.00, -0.07, -0.18, -0.35, -0.63, -1.10,
* -0.00, -0.10, -0.23, -0.45, -0.80, -1.40,
* -0.00, -0.08, -0.20, -0.40, -0.70, -1.27,
* -0.00, -0.06, -0.15, -0.30, -0.60, -1.13/
 DATA CN4/
*  0.00,  0.05,  0.15,  0.43,  0.97,  1.85,
*  0.00,  0.05,  0.12,  0.48,  1.05,  1.92,
*  0.00,  0.07,  0.23,  0.55,  1.17,  2.12,
*  0.00,  0.10,  0.30,  0.80,  1.65,  3.05,
*  0.00,  0.06,  0.25,  0.70,  1.55,  2.95,
*  0.00,  0.04,  0.20,  0.65,  1.50,  2.85/
 DATA CL4/
* .000 , .002 , .005 , .016 , .036 , .067 , .110 ,
* .000 , .001 , .002 , .006 , .018 , .042 , .020 ,
* .000 , .000 , .000 , .003 , .013 , .009 ,-.013 /
 DATA CL2/
* .000 , .005 , .013 , .034 , .066 , .062 , .010 ,
* .000 , .004 , .010 , .034 , .080 , .086 , .055 ,
* .000 , .003 , .007 , .023 , .044 , .030 ,-.010 /
 DATA CZ2/
*  .00 ,  .02 ,  .06 ,  .25 ,  .55 , 1.00 , 1.58 ,
*  .00 ,  .02 ,  .06 ,  .25 ,  .58 , 1.04 , 1.63 ,
*  .00 ,  .02 ,  .06 ,  .25 ,  .55 , 1.00 , 1.58 ,
*  .00 ,  .02 ,  .06 ,  .25 ,  .55 , 1.00 , 1.54 ,
*  .00 ,  .02 ,  .06 ,  .25 ,  .52 ,  .83 , 1.12 /
 DATA CM2/
*-0.00 ,-0.05 ,-0.10 ,-0.40 ,-1.05 ,-2.05 ,-3.40 ,
*-0.00 ,-0.05 ,-0.10 ,-0.40 ,-1.05 ,-2.20 ,-3.75 ,
*-0.00 ,-0.00 ,-0.05 ,-0.30 ,-0.95 ,-2.10 ,-3.50 ,
*-0.00 ,-0.05 ,-0.10 ,-0.45 ,-1.15 ,-2.45 ,-3.90 ,
*-0.00 ,-0.05 ,-0.10 ,-0.55 ,-1.35 ,-2.40 ,-3.6 /
 DATA CMQ/
* -1.75, -2.74, -3.06, -3.07, -2.88, -2.55,
* -1.75, -2.73, -3.02, -3.03, -2.87, -2.57,
* -1.75, -2.68, -2.98, -3.00, -2.88, -2.63,
* -1.75, -2.74, -3.14, -3.30, -3.22, -3.26,
* -1.75, -2.74, -3.14, -3.30, -3.25, -3.34,
* -1.75, -2.79, -3.20, -3.38, -3.50, -3.59/
 DATA CLP/
* -.038, -.057, -.072, -.079, -.081, -.076,
* -.038, -.057, -.072, -.079, -.081, -.077,
* -.038, -.057, -.072, -.079, -.082, -.079,
* -.042, -.061, -.078, -.093, -.106, -.117,
* -.041, -.059, -.076, -.090, -.103, -.113,
* -.038, -.053, -.068, -.083, -.099, -.114/
 DATA NC6/6,4,24,0/
 DATA BA6/0.,2.,4.,8.,12.,16./
 DATA VM6/.85,.95,1.05,1.25/
 DATA DXCPF/
*.0000,.0000,.0000,.0000,.0000,.0000,
*.0000,.0000,.0000,.0055,.0075,.0085,
*.0140,.0135,.0125,.0120,.0115,.0110,
*.0415,.0410,.0390,.0280,.0190,.0150/
 DATA NC7/9,3,27,0/
 DATA BA7/0.0,2.0,4.1,6.1,8.1,10.2,13.2,16.3,19.3/
 DATA VM7/.5,.95,1.25/
 DATA DCMF/
```

```
*.0139,.0494,.1190,.2259,.2856,.3479,.4598,.4968,.6577,
* .0510,.0848, .1798,.2990,.3764,.4833,.5519,.6117,.7500,
*.0276,.0826,.1630,.2792,.3381,.4157,.4995,.5436,.5379/
 DATA NC8/3,0/
 DATA VM8/.5,.95,1.25/
 DATA DCMO/.0139,.0510,.0276/
 END
```

```
      SUBROUTINE C1
      COMMON C(3830)
      DIMENSION BDELTC(4),VAR(101)
C
C**INPUT DATA
      EQUIVALENCE (C( 860),TDY    )
      EQUIVALENCE (C( 861),GBIAS  )
      EQUIVALENCE (C( 862),GN     )
      EQUIVALENCE (C( 863),WN2    )
      EQUIVALENCE (C( 864),WN1    )
      EQUIVALENCE (C( 865),WL     )
      EQUIVALENCE (C( 866),WLXX1  )
      EQUIVALENCE (C( 867),WLXX2  )
      EQUIVALENCE (C( 868),WLJK1  )
      EQUIVALENCE (C( 869),WLJK2  )
      EQUIVALENCE (C( 870),HJK    )
      EQUIVALENCE (C( 871),WXX    )
      EQUIVALENCE (C( 872),DXX    )
      EQUIVALENCE (C( 873),WJK    )
      EQUIVALENCE (C( 874),DJ    )
      EQUIVALENCE (C( 875),GXX    )
      EQUIVALENCE (C( 876),GJK    )
      EQUIVALENCE (C( 877),RES    )
      EQUIVALENCE (C( 878),QDN    )
      EQUIVALENCE (C( 879),QUP    )
      EQUIVALENCE (C( 890),HXX    )
      EQUIVALENCE (C( 892),QBIAS  )
      EQUIVALENCE (C( 893),RBIAS  )
      EQUIVALENCE (C( 899),OPTC1  )
      EQUIVALENCE (C( 947),GNS    )
      EQUIVALENCE (C( 948),WS1    )
      EQUIVALENCE (C( 949),WS2    )
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(  77),SPHI    )
      EQUIVALENCE (C(  87),STHT    )
      EQUIVALENCE (C(  97),SPSI    )
      EQUIVALENCE (C( 353),BPH1   )
      EQUIVALENCE (C( 354),BTH2   )
      EQUIVALENCE (C( 355),BPS1   )
      EQUIVALENCE (C( 403),WLAMQ  )
      EQUIVALENCE (C( 407),WLAMR  )
      EQUIVALENCE (C( 461),CAGE   )
      EQUIVALENCE (C( 462),TKRZ   )
      EQUIVALENCE (C( 463),TKRY   )
      EQUIVALENCE (C(1233),BDR    )
      EQUIVALENCE (C(1747),WR     )
      EQUIVALENCE (C(1743),WQ     )
      EQUIVALENCE (C(1739),WP     )
C
C**INPUTS FROM MAIN PROGRAM
      EQUIVALENCE (C(2000),T      )
C
C** STATE VARIABLE OUTPUTS
      EQUIVALENCE (C( 800),WLQSDD)
      EQUIVALENCE (C( 803),WLQSP )
      EQUIVALENCE (C( 804),WLQSD )
      EQUIVALENCE (C( 807),WLQS  )
      EQUIVALENCE (C( 808),WLQSSD)
```

```
      EQUIVALENCE (C( 811),WLQSS )
      EQUIVALENCE (C( 812),WLRSDD)
      EQUIVALENCE (C( 815),WLRSP )
      EQUIVALENCE (C( 816),WLRSD )
      EQUIVALENCE (C( 819),WLRS  )
      EQUIVALENCE (C( 820),WLRSSD)
      EQUIVALENCE (C( 823),WLRSS )
      EQUIVALENCE (C( 824),BLQSSD)
      EQUIVALENCE (C( 827),BLQSS )
      EQUIVALENCE (C( 828),BLRSSD)
      EQUIVALENCE (C( 831),BLRSS )
      EQUIVALENCE (C( 832),BJJSDD)
      EQUIVALENCE (C( 835),BJJSP )
      EQUIVALENCE (C( 836),BJJSD )
      EQUIVALENCE (C( 839),BJJS  )
      EQUIVALENCE (C( 840),BKKSDD)
      EQUIVALENCE (C( 843),BKKSP )
      EQUIVALENCE (C( 844),BKKSD )
      EQUIVALENCE (C( 847),BKKS  )
      EQUIVALENCE (C( 848),BXXSDD)
      EQUIVALENCE (C( 851),BXXSP )
      EQUIVALENCE (C( 852),BXXSD )
      EQUIVALENCE (C( 855),BXXS  )
      EQUIVALENCE (C( 931),BJSSD ), (C( 934),BJSS  )
      EQUIVALENCE (C( 935),BKSSD ), (C( 938),BKSS  )
      EQUIVALENCE (C( 950),SNP2  ), (C( 953),SNP1  ),(C( 956),SNP0  )
      EQUIVALENCE (C( 957),SNQ2  ), (C( 960),SNQ1  ),(C( 963),SNQ0  )
      EQUIVALENCE (C( 964),SNR2  ), (C( 967),SNR1  ),(C( 970),SNR0  )
      EQUIVALENCE (C( 971),BPC2  ), (C( 974),BPC1  ),(C( 977),BPC0  )
      EQUIVALENCE (C(903),H13P),(C(904),H13M)
      EQUIVALENCE (C(905),H24P),(C(906),H24M)
      EQUIVALENCE (C(907),CDRFT1),(C(908),CDRFT2)
      EQUIVALENCE (C(909),CDRFTY)
      EQUIVALENCE (C(984),CDRFTX)
      EQUIVALENCE (C(1676),ANGX)
      EQUIVALENCE (C(978),BDRFTD),(C(981),BDRFT)
      EQUIVALENCE (C(985),NLMT1),(C(986),NLMT2)
      EQUIVALENCE (C(987),BJJSSS),(C(988),BKKSSS)
      EQUIVALENCE (C(989),BXXSSS)
      EQUIVALENCE (C(990),BJJSSL),(C(991),BKKSSL)
      EQUIVALENCE (C(530),BJSDD),(C(534),BJSD),(C(537),BJS)
      EQUIVALENCE (C(538),BKSDD),(C(542),BKSD),(C(545),BKS)
      EQUIVALENCE (C(546),BXSDD),(C(550),BXSD),(C(553),BXS)
      EQUIVALENCE (C(533),CJSD),(C(541),CKSD),(C(549),CXSD)
       EQUIVALENCE (C(942),WL2)
       EQUIVALENCE (C(943),DJ2)
       EQUIVALENCE (C(944),WJ2)
       EQUIVALENCE (C(945),DX2)
       EQUIVALENCE (C(946),WX2)
      EQUIVALENCE (C(2965),VAR(1))
C
C**OUTPUTS
      EQUIVALENCE (C( 856),BDELTC(1))
C
CM*OTHER OUTPUTS
      EQUIVALENCE (C( 880),BPHIS )
      EQUIVALENCE (C(518),B13SS)
      EQUIVALENCE (C(519),B24SS)
      EQUIVALENCE (C( 881),BJJ   )
      EQUIVALENCE (C( 882),BKK   )
```

```
      EQUIVALENCE (C( 883),BXXSS )
      EQUIVALENCE (C( 884),BJJSS )
      EQUIVALENCE (C( 885),BKKSS )
      EQUIVALENCE (C( 886),BTHTS )
      EQUIVALENCE (C( 887),BPSIS )
C
C**GUIDANCE SIGNAL SHAPING
C**GUIDANCE SWITCHING
      WLQSD = WLQSP
      WLRSD = WLRSP
      WLQSDD = WN2*(WN2*(WLAMQ - WLQS) - 2.*WLQSD)
      WLRSDD = WN2*(WN2*(WLAMR - WLRS) - 2.*WLRSD)
      WQC = GN*(WLQSD/WL + WLQS) + QBIAS + GBIAS
      WRC = GN*(WLRSD/WL+WLRS) + RBIAS
      IF (TKRZ.GT.0. .AND. T.GT.TDY) GO TO 4
      WLQSDD = 0.
      WQC = QBIAS + GBIAS + QDN
      IF(CAGE .GT. 0. .AND. T.GT. TDY) WQC = WLAMQ + QBIAS + GBIAS
    4 IF (TKRY .GT. 0.) GO TO 5
      WLRSDD = 0.
      WRC = RBIAS
      IF(CAGE .GT. 0.) WRC = WLAMR + RBIAS
    5 CONTINUE
      WLQSSD = WN1*(WQC - WLQSS)
      WLRSSD = WN1*(WRC - WLRSS)
      IF(WN1 .GT. 0.) GO TO 3
      WLQSS = WQC
      WLRSS = WRC
    3 BLQSSD = WLQSS
      BLRSSD = WLRSS
C
C**RATE GYRO DYNAMICS AND LIMITING
      BDRFTD=(CDRFT1*ANGX+CDRFT2)
      BTHTS=-BTH2+BDRFT
      BPSIS=-BPS1+CDRFTY*BDRFT
      BPHIS=-BPH1+CDRFTX*BDRFT
      BPHISD = WP - (WQ*COSD(-BPH1) -WR*SIND(-BPH1))
     *        *SIND(-BPS1)/COSD(-BPS1)
      BPHS = BPHISD / WLXX2 + BPHIS
    6 IF(GNS .LE. 0.) GO TO 8
      SNP2 = WS1*WS2*(GNS*SPHI-SNP0) - (WS1+WS2)*SNP1
      SNQ2 = WS1*WS2*(GNS*STHT-SNQ0) - (WS1+WS2)*SNQ1
      SNR2 = WS1*WS2*(GNS*SPSI-SNR0) - (WS1+WS2)*SNR1
      BPHS = BPHS -SNP1
      BTHTS = BTHTS - SNQ1
      BPSIS = BPSIS - SNR1
    8 CONTINUE
      BXX = BPHS
      BTSS = BTHTS
      BPSS = BPSIS
C*****SPECIAL CASE - PROGRAMMED FLIGHT
      IF(OPTC1 .LE. 0.) GO TO 9
      BLQSSD = QBIAS
      BLRSSD = 0.
      BDC = 0.
      BT=0.
      BP=0.
      IF(T.GE.1.0000 .AND. T.LE.3.2000)BDC= 5.
      IF(T.GE.4.2    .AND. T.LE.6.4000)BT= 5.
      IF(T.GE.7.6000 .AND. T.LE.9.9000)BDC=-5.
```

```
      IF(T.GE.11.000 .AND. T.LE.13.100)BP=-5.
      IF(T.GE.15.200 .AND. T.LE.17.300)BDC= 5.
      IF(T.GE.18.320 .AND. T.LE.20.560)BT = 5.
      IF(T.GE.21.545 .AND. T.LE.23.675)BDC=-5.
      IF(T.GE.24.770 .AND. T.LE.26.910)BP=-5.
      C(520)=-BPCO+BP-BT
       BPC2=18.18*20.57*(BDC-BPCO)-(18.18+20.57)*BPC1
      BXX = BXX + BPCO
      BTSS=BTSS-BT
       BPSS=BPSS-BP
C
C**SUMMATION OF RATE DAMPING AND GUIDANCE SIGNALS AND THEIR DERIVATIVES
    9 BJJ = (BLRSS - BPSS) - (BLQSS - BTSS)
      BKK = -(BLRSS-BPSS) - (BLQSS-BTSS)
C
C**GUIDANCE SIGNAL SHAPING AND LIMITING
      BXXSD = BXXSP
      BJJSD = BJJSP
      BKKSD = BKKSP
      BXXSDD = WXX*(WXX*(BXX - BXXS) - 2.*DXX*BXXSD)
      BJSD=CJSD
      BKSD=CKSD
      BXSD=CXSD
      HSAT=23.
      IF(ABS(BJJS).LT.HSAT)GO TO 20
      BJJS=SIGN(HSAT,BJJS)
      VAR(NLMT1+1)=BJJS
      IF(BJJS*BJJSD.GT.0.0)BJJSD=0.0
 20   BJJSDD=WJK*(WJK*(GJK*BJJ-BJJS)-2.*DJK*BJJSD)
      IF(ABS(BKKS).LT.HSAT)GO TO 25
      BKKS=SIGN(HSAT,BKKS)
      VAR(NLMT1+3)=BKKS
      IF(BKKS*BKKSD.GT.0.0)BKKSD=0.0
 25   BKKSDD=WJK*(WJK*(GJK*BKK-BKKS)-2.*DJK*BKKSD)
      BXXSS=GXX*((BXXSDD+(WLXX1+WLXX2)*BXXSD)/(WLXX1*WLXX2)+BXXS)
      BJJSS=BJJSD/WLJK1+BJJS
      IF(ABS(BJJSS).GT.HSAT)BJJSS=SIGN(HSAT,BJJSS)
      BKKSS=BKKSD/WLJK1+BKKS
      IF(ABS(BKKSS).GT.HSAT)BKKSS=SIGN(HSAT,BKKSS)
C **  HIGH FREQUENCY SHAPINF OPTION
      IF(WX2*WJ2.LE.0.)GO TO 10
      IF(ABS(BKS).LT.HSAT)GO TO 30
      BKS=SIGN(HSAT,BKS)
      VAR(NLMT2+3)=BKS
      IF(BKS*BKSD.GT.0.0)BKSD=0.0
 30    BKSDD=WJ2*(WJ2*(BKKSS-BKS)- 2.*DJ2*BKSD)
      IF(ABS(BJS).LT.HSAT)GO TO 35
      BJS=SIGN(HSAT,BJS)
      VAR(NLMT2+1)=BJS
      IF(BJS*BJSD.GT.0.0)BJSD=0.0
 35    BJSDD=WJ2*(WJ2*(BJJSS-BJS)-  2.*DJ2*BJSD)
      BXSDD=WX2*(WX2*(BXXSS-BXS)-2.*DX2*BXSD)
      BKKSSS=BKSD/WLJK2+BKS
      IF(ABS(BKKSSS).GT.HSAT)BKKSSS=SIGN(HSAT,BKKSSS)
      BJJSSS=BJSD/WLJK2+BJS
      IF(ABS(BJJSSS).GT.HSAT)BJJSSS=SIGN(HSAT,BJJSSS)
      BXXSSS=BXS
 10   BJJSSL=BJJSSS
       BKKSSL=BKKSSS
      IF(BJJSSL.GT.H13P)BJJSSL=H13P
```

```
      IF(BJJSSL.LT.H13M)BJJSSL=H13M
      IF(BKKSSL.GT.H24P)BKKSSL=H24P
      IF(BKKSSL.LT.H24M)BKKSSL=H24M
      B13SS=BJJSSL
      B24SS=BKKSSL
C
C**COMMANDS TO ACTUATORS
      BJSSD = 20.*(B13SS-BJSS)
      BKSSD = 20.*(B24SS-BKSS)
      B13SS = BJSSD/125.+BJSS
      B24SS = BKSSD/125.+BKSS
      BDELTC(1)=B13SS+BXXSSS
      BDELTC(2)=B24SS+BXXSSS
      BDELTC(3)=B13SS-BXXSSS
      BDELTC(4)=B24SS-BXXSSS
      RETURN
      END
```

```
      SUBROUTINE C1I
      COMMON C(3830)
      DIMENSION IPL(100)
      EQUIVALENCE (C(2561),N      )
      EQUIVALENCE (C(2562),IPL(1)    )
      EQUIVALENCE (C(985),NLMT1),(C(986),NLMT2)
C
      IPL(N   ) = 800
      IPL(N+1) = 804
      IPL(N+2) = 808
      IPL(N+3) = 812
      IPL(N+4) = 816
      IPL(N+5) = 820
      IPL(N+6) = 824
      IPL(N+7) = 828
      IPL(N+8) = 832
      NLMT1=N+9
      IPL(N+9) = 836
      IPL(N+10) = 840
      IPL(N+11) = 844
      IPL(N+12) = 848
      IPL(N+13) = 852
      N = N+14
      C(803) = 0.
      C(807) = 0.
      C(811) = 0.
      C(815) = 0.
      C(819) = 0.
      C(823) = 0.
      C(827) = 0.
      C(831) = 0.
      C(835) = 0.
      C(839) = 0.
      C(843) = 0.
      C(847) = 0.
      C(851) = 0.
      C(855) = 0.
      C( 811) = C( 878) + C (861)
      C( 823) = C( 879)
      C( 831) = -C(1143)
      C( 847) =-C( 831)
      C( 839) = C( 831)
CCC   PROGRAMMER
      IF(C(899).LE.0.)GO TO 9
      IPL(N)=971
      IPL(N+1)=974
      N=N+2
      C(974)=0.
      C(977)=0.
 9    IF(C(949) .LE. 0.) GO TO 10
C**GYRO POT NOISE
      IPL(N   ) = 950
      IPL(N+1) = 953
      IPL(N+2) = 957
      IPL(N+3) = 960
      IPL(N+4) = 964
      IPL(N+5) = 967
      N = N + 6
      C( 953) = 0.
```

```
      C( 956) = 0.
      C( 960) = 0.
      C( 963) = 0.
      C( 967) = 0.
      C( 970) = 0.
   10 CONTINUE
      IF(C(944)*C(946).LE.0.0)GO TO 20
      IPL(N)=530
      IPL(N+1)=534
      NLMT2=N+1
      IPL(N+2)=538
      IPL(N+3)=542
      IPL(N+4)=546
      IPL(N+5)=550
      N=N+6
      C(533)=0.0
      C(541)=0.0
      C(549)=0.0
      C(537)=0.0
      C(545)=0.0
      C(553)=0.0
20      CONTINUE
      IPL(N  ) = 931
      IPL(N+1) = 935
      N = N + 2
      C( 934) = 0.
      C( 938) = 0.
      IPL(N)=978
      N=N+1
      C(981)=0.0
      RETURN
      END
```

```
      SUBROUTINE C4
C
C**  HELFIRE SIMPLIFIED ACTUATOR MODEL
C****** NON - LINEAR MODEL ******
C
      COMMON C(3830)
       DIMENSION BDSS(4),BDS(4),BDSD(4)
      DIMENSION BDELTD(4),BDELT(4),BDELTC(4),VAR(101)
      DIMENSION BDLT(4), BDT(4)
      DIMENSION  WDS2D(4), WDS2(4), WDS1D(4), WDS1(4)
      DIMENSION IPL(101)
      DIMENSION NC2(2), CB2(6), CHAF(6)
      DIMENSION  G1(4),G2(4),G3(4),W1(4),ZN(4),WN(4)
      DIMENSION H1(4), H2(4), BH(4)
      DIMENSION CB1(6), CHDU(6), CHDL(6)
        DIMENSION AIH(4)
      EQUIVALENCE (C(521),AIH(1))
       EQUIVALENCE (C(1116),G3(1))
      EQUIVALENCE (C(1120),G1(1))
      EQUIVALENCE (C(1124),ZN(1))
      EQUIVALENCE (C(1128),WN(1))
      EQUIVALENCE (C(1132),W1(1))
      EQUIVALENCE (C(1136),G2(1))
      EQUIVALENCE (C(1148),H1(1))
      EQUIVALENCE (C(1152),H2(1))
      EQUIVALENCE (C(1156),BH(1))
C
C**INPUT DATA
      EQUIVALENCE (C(1140),OPTACT)
      EQUIVALENCE (C(1141),BDP    )
      EQUIVALENCE (C(1142),BDQ    )
      EQUIVALENCE (C(1143),BDR    )
      EQUIVALENCE (C(1144),EFF    )
      EQUIVALENCE (C(1145),HX     )
      EQUIVALENCE (C(1146),BDB    )
      EQUIVALENCE (C(1147),FMBS   )
      EQUIVALENCE (C(1306),RFAREA)
      EQUIVALENCE (C(1307),RFLGTH)
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(0203),PDYNMC)
      EQUIVALENCE (C( 204),VMACH )
      EQUIVALENCE (C( 367),BALPHA)
      EQUIVALENCE (C( 368),BALPHY)
      EQUIVALENCE (C( 377),BALPD )
      EQUIVALENCE (C( 378),BALYD )
      EQUIVALENCE (C( 856),BDELTC(1))
      EQUIVALENCE (C(1254), DELTB)
      EQUIVALENCE (C(1192),BDT(1))
      EQUIVALENCE (C(1196),BDLT(1))
       EQUIVALENCE (C(525),BDSS(1))
      EQUIVALENCE (C(1310),FMH2  )
      EQUIVALENCE (C(1311),FMH3  )
      EQUIVALENCE (C(1312),FMH4  )
C
C**INPUTS FROM MAIN PROGRAM
      EQUIVALENCE (C(2000),T      )
      EQUIVALENCE (C(2013),DOC    )
      EQUIVALENCE (C(2561),N      )
```

```
      EQUIVALENCE (C(2562),IPL(1))
C
      DATA WATLB1/4HCH14/ ,WATLB2/4HCH23/ ,WATLB3/4HFMHA/
      DATA NC2/6,0/
      DATA CB2/  .00,  .50,  .85, 1.05, 1.40, 1.60/
      DATA CHAF/ 0. ,   .1,   .4,  1.6,  3.1,  3.6/
      DATA  CB1/.0000,.8500,.9500,1.050,1.250,1.600/
      DATA CHDU/.0007,.0007,.0007,.0020,.0020,.0020/
      DATA CHDL/.0007,.0007,.0018,.0018,.0018,.0018/
C**STATE VARIABLE OUTPUTS
      BDELTC(1) = BDELTC(1) - BDP + BDQ - BDR
      BDELTC(2) = BDELTC(2) - BDP + BDQ + BDR
      BDELTC(3) = BDELTC(3) + BDP + BDQ - BDR
      BDELTC(4) = BDELTC(4) + BDP + BDQ + BDR
      BDELT(1)=C(1103)
      BDELT(2)=C(1107)
      BDELT(3)=C(1111)
      BDELT(4)=C(1115)
      BDS(1)=C(1087)
      BDS(2)=C(1091)
      BDS(3)=C(1095)
      BDS(4)=C(1099)
      WDS2 (1) = C(1163)
      WDS2 (2) = C(1167)
      WDS2 (3) = C(1171)
      WDS2 (4) = C(1175)
      WDS1 (1) = C(1179)
      WDS1 (2) = C(1183)
      WDS1 (3) = C(1187)
      WDS1 (4) = C(1191)
C
C**ACTUATOR DYNAMICS
      XK=0.
      VM = AMIN1(VMACH,1.4)
      CALL TABLE(VM,CB1,CHDU,NC2(1),XK,WATLB1,CH14)
      CALL TABLE(VM,CB1,CHDL,NC2(1),XK,WATLB2,CH23)
      CALL TABLE(VM,CB2,CHAF,NC2(1),XK,WATLB3,FMHA)
      FM14 = CH14*PDYNMC*RFLGTH*RFAREA*12.
      FM23 = CH23*PDYNMC*RFLGTH*RFAREA*12.
      DO 30  I=1,4
      J = (I+1)/2
      FMHD = FM14
      IF(IABS((2*I-5)/2) .LE. 0) FMHD = FM23
      FMH=FMHD*BDELT(I)+FMHA*(BALPHA+BALPHY*(-1)**I)
      BDE=WDS1(I)-FMH/G2(I)
      WDS2D(I)=WN(I)*(G3(I)*BDE-WDS2(I))
      BDELTD(I)=WDS2D(I)/W1(I)+WDS2(I)
      BDSD(I)=65.*(BDELTD(I)/125.+BDELT(I)-BDS(I))
      BDSS(I)=BDSD(I)/125.+BDS(I)
      BDH=G1(I)*(BDELTC(I)-BDSS(I))
      AIH(I)=BDH
       IF(BDH.LT.-H2(I))BDH=-H2(I)
       IF(BDH.GT. H1(I))BDH= H1(I)
      WDS1D(I)=W1(I)*(BDH-WDS1(I))
      IF(BDT(I).LT.BDELT(I)-BH(I))BDT(I)=BDELT(I)-BH(I)
      IF(BDT(I).GT.BDELT(I)+BH(I))BDT(I)=BDELT(I)+BH(I)
      BDLT(I)=BDT(I)
C**SURFACE POSITION LIMITER
      IF((ABS(BDELT(I)).GT.19.).AND.(BDELTD(I)*BDELT(I).GT.0.))BDELTD(I)
     *=0.
```

```
   30 CONTINUE
C
        C(1084)=BDSD(1)
        C(1088)=BDSD(2)
        C(1092)=BDSD(3)
        C(1096)=BDSD(4)
      C(1103) = BDELT(1)
      C(1107) = BDELT(2)
      C(1111) = BDELT(3)
      C(1115) = BDELT(4)
C
C**OUTPUT DERIVATIVES OF STATE VARIABLES TO INTEGRATION
      C(1100) = BDELTD(1)
      C(1104) = BDELTD(2)
      C(1108) = BDELTD(3)
      C(1112) = BDELTD(4)
      C(1160) = WDS2D(1)
      C(1164) = WDS2D(2)
      C(1168) = WDS2D(3)
      C(1172) = WDS2D(4)
      C(1176) = WDS1D(1)
      C(1180) = WDS1D(2)
      C(1184) = WDS1D(3)
      C(1188) = WDS1D(4)
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      IF (T.GT.0. .OR. DOC.GE.6.) RETURN
      WRITE(6,150) T
      DO 100 I=2,N
      J = IPL(I-1)
  100 WRITE(6,200) J, C(J), C(J+3)
      CALL DUMPO
  150 FORMAT(F10.4)
  200 FORMAT(I10,1P2E15.7)
C
      RETURN
      END
```

```
SUBROUTINE C4I
COMMON C(3830)
DIMENSION IPL(100), ISNDX(40)
EQUIVALENCE (C(3634), ISNDX(1)), (C(3512), I3512)
DIMENSION BDLT(4)
DIMENSION  G1(4),G2(4),G3(4),W1(4),ZN(4),WN(4)
DIMENSION H1(4), H2(4), BH(4)
EQUIVALENCE (C(1120),G1(1) )
EQUIVALENCE (C(1124),ZN(1) )
EQUIVALENCE (C(1128),WN(1) )
EQUIVALENCE (C(1132),W1(1) )
EQUIVALENCE (C(1136),G2(1) )
EQUIVALENCE (C(1116),G3(1))
EQUIVALENCE (C(1148),H1(1) )
EQUIVALENCE (C(1152),H2(1) )
EQUIVALENCE (C(1156),BH(1) )
EQUIVALENCE (C(1103),BDELT1)
EQUIVALENCE (C(1107),BDELT2)
EQUIVALENCE (C(1111),BDELT3)
EQUIVALENCE (C(1115),BDELT4)
EQUIVALENCE (C(1247),FELECB )
EQUIVALENCE (C(1248),FELECQ)
EQUIVALENCE (C(1249),FELECR)
EQUIVALENCE (C(1250),FMECHB )
EQUIVALENCE (C(1251),FMECHQ)
EQUIVALENCE (C(1252),FMECHR)
EQUIVALENCE (C(1140),OPTACT)
EQUIVALENCE (C(1141),BDP    )
EQUIVALENCE (C(1142),BDQ    )
EQUIVALENCE (C(1143),BDR    )
EQUIVALENCE (C(1196),BDLT(1))
EQUIVALENCE (C(2561),N      )
EQUIVALENCE (C(2562),IPL(1))
C(377)=0.0
C(378)=0.0
C(1192)=0.0
C(1193)=0.0
C(1194)=0.0
C(1195)=0.0
IPL(N) = 1100
IPL(N+1) = 1104
IPL(N+2) = 1108
IPL(N+3) = 1112
N = N+4
IPL(N  ) = 1160
IPL(N+1) = 1164
IPL(N+2) = 1168
IPL(N+3) = 1172
IPL(N+4) = 1176
IPL(N+5) = 1180
IPL(N+6) = 1184
IPL(N+7) = 1188
N = N+8
IPL(N)=1084
IPL(N+1)=1088
IPL(N+2)=1092
IPL(N+3)=1096
N=N+4
 C(1087)=0.0
```

```
        C(1091)=0.0
        C(1095)=0.0
        C(1099)=0.0
       C(1163) = 0.
       C(1167) = 0.
       C(1171) = 0.
       C(1175) = 0.
       C(1179) = 0.
       C(1183) = 0.
       C(1187) = 0.
       C(1191) = 0.
       RETURN
C
       ENTRY A1I
C
C MONTE CARLO FIN MISALIGNMENT ERRORS
C
       FELECB  = 0.
       FELECQ = 0.
       FELECR = 0.
       FMECHB  = 0.
       FMECHQ = 0.
       FMECHR = 0.
       DO 10 I = 1, I3512
       IDO = I
       IF(ISNDX(I).EQ.1250) CALL MCARLO (DUM, 1, IDO)
       IF(ISNDX(I).EQ.1251) CALL MCARLO (DUM, 1, IDO)
       IF(ISNDX(I).EQ.1252) CALL MCARLO (DUM, 1, IDO)
C MONTE CARLO FIN OFFSET (MODULE C4I AND C4)
       IF(ISNDX(I).EQ.1247) CALL MCARLO (DUM, 1, IDO)
       IF(ISNDX(I).EQ.1248) CALL MCARLO (DUM, 1, IDO)
       IF(ISNDX(I).EQ.1249) CALL MCARLO (DUM, 1, IDO)
       DELTB  = FELECB  + FMECHB
       DELTQB = FELECQ + FMECHQ
       DELTRB = FELECR + FMECHQ
    10 CONTINUE
       IF (OPTACT .LE. 0.) GO TO 20
       OPTACT = 0.
       DO 5 I=1,4
       READ(5,200)G1(I),ZN(I),WN(I),W1(I),G3(I),G2(I),
      * H1(I),H2(I),BH(I),I8
 5     WRITE(6,300)I8,I,G1(I),ZN(I),WN(I),W1(I),G3(I),G2(I),
      *   H1(I),H2(I),BH(I)
   200 FORMAT(9F8.3,I8)
   300 FORMAT(I6,I2,9F8.3)
    20 CONTINUE
       BDELT1 = -BDP + BDQ - BDR
       BDELT2 = -BDP + BDQ + BDR
       BDELT3 =  BDP + BDQ - BDR
       BDELT4 =  BDP + BDQ + BDR
       BDLT(1) = BDELT1
       BDLT(2) = BDELT2
       BDLT(3) = BDELT3
       BDLT(4) = BDELT4
       RETURN
       END
```

```
      SUBROUTINE   DTA(XIN,J)
      COMMON C(3830)
      DIMENSION REF(4),BIT(4)
      EQUIVALENCE (C(3453),REF(1))
      EQUIVALENCE (C(3457),BIT(1))
C     REF(1 THRU 4) IS BDELTC(1 THRU 4)
      SIG =1.
      IF(XIN.LT.0.)SIG =-1.
      XIN=ABS(XIN)
      IF(XIN.LT.REF(  J))GO TO 10
      WRITE(6,14)C(2000),XIN,J
 14   FORMAT(6H DTOA ,2E20.10,I5)
      XIN =REF(J)
      GO TO 100
 10   XIN=XIN-AMOD(XIN,BIT(J))
 100  XIN=SIG *XIN
       RETURN
      END
```

```
      SUBROUTINE DUMPO
      COMMON C(3830)
      DO 100 I=1, 1800, 7
      N = 0
      DO 200 J=1, 7
      K = I + J - 1
  200 IF (ABS(C(K)) .GT. 1.E-10) N = 1
  100 IF (N .GT. 0) WRITE(6,300)
     *     I,C(I),C(I+1),C(I+2),C(I+3),C(I+4),C(I+5),C(I+6)
  300 FORMAT(1H ,I5,1P7E15.7)
      RETURN
      END
```

```
      SUBROUTINE D1
C**TRANSLATIONAL DYNAMICS MODULE
      COMMON C(3830)
C
C**INPUT DATA
      EQUIVALENCE (C(1627),AGRAV )
      EQUIVALENCE (C(1628),DMASS )
      EQUIVALENCE (C(1629),ATHRST)
      EQUIVALENCE (C(1630),ATURNT)
      EQUIVALENCE (C(1631),BGAMT )
      EQUIVALENCE (C(1639),OPTARG)
      EQUIVALENCE (C(1681),ADIVE )
      EQUIVALENCE (C(1751),CRAD  )
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(1300),FXBA  )
      EQUIVALENCE (C(1301),FYBA  )
      EQUIVALENCE (C(1302),FZBA  )
      EQUIVALENCE (C(1703),CFA11 )
      EQUIVALENCE (C(1707),CFA12 )
      EQUIVALENCE (C(1711),CFA13 )
      EQUIVALENCE (C(1715),CFA21 )
      EQUIVALENCE (C(1719),CFA22 )
      EQUIVALENCE (C(1723),CFA23 )
      EQUIVALENCE (C(1727),CFA31 )
      EQUIVALENCE (C(1731),CFA32 )
      EQUIVALENCE (C(1735),CFA33 )
      EQUIVALENCE (C(2000),T     )
C
C**STATE VARIABLE OUTPUTS
      EQUIVALENCE (C(1600),VXED  )
      EQUIVALENCE (C(1603),VXE   )
      EQUIVALENCE (C(1604),VYED  )
      EQUIVALENCE (C(1607),VYE   )
      EQUIVALENCE (C(1608),VZED  )
      EQUIVALENCE (C(1611),VZE   )
      EQUIVALENCE (C(1612),RXED  )
      EQUIVALENCE (C(1615),RXE   )
      EQUIVALENCE (C(1616),RYED  )
      EQUIVALENCE (C(1619),RYE   )
      EQUIVALENCE (C(1620),RZED  )
      EQUIVALENCE (C(1623),RZE   )
      EQUIVALENCE (C(1640),VTARGD)
      EQUIVALENCE (C(1643),VTARG )
      EQUIVALENCE (C(1644),BPSITD)
      EQUIVALENCE (C(1647),BPSIT )
      EQUIVALENCE (C(1648),RTXED )
      EQUIVALENCE (C(1651),RTXE  )
      EQUIVALENCE (C(1652),RTYED )
      EQUIVALENCE (C(1655),RTYE  )
      EQUIVALENCE (C(1656),RTZED )
      EQUIVALENCE (C(1659),RTZE  )
C
C**OTHER OUTPUTS
      EQUIVALENCE (C(1624),AXBA  )
      EQUIVALENCE (C(1625),AYBA  )
      EQUIVALENCE (C(1626),AZBA  )
      EQUIVALENCE (C(1632),VDELX )
      EQUIVALENCE (C(1633),VDELY )
```

```
      EQUIVALENCE (C(1634),VDELZ )
      EQUIVALENCE (C(1635),RDELX )
      EQUIVALENCE (C(1636),RDELY )
      EQUIVALENCE (C(1637),RDELZ )
      EQUIVALENCE (C(1638),VCLSNG)
      EQUIVALENCE (C(1660),VTXE  )
      EQUIVALENCE (C(1661),VTYE  )
      EQUIVALENCE (C(1662),VTZE  )
      EQUIVALENCE (C(1663),VDXB  )
      EQUIVALENCE (C(1664),VDYB  )
      EQUIVALENCE (C(1665),VDZB  )
      EQUIVALENCE (C(1676),ANGX  )
      EQUIVALENCE (C(1677),ANGY  )
      EQUIVALENCE (C(1678),ANGZ  )
      EQUIVALENCE (C( 371),RANGE   )
C
C**ADD AERO AND THRUST FORCES TO GET TOTAL ACCELERATION IN BODY AXES
      AXBA = FXBA/DMASS
      AYBA = FYBA/DMASS
      AZBA = FZBA/DMASS
C
C**RESOLVE FROM BODY TO EARTH AXES
      AXE  = CFA11*AXBA+CFA21*AYBA+CFA31*AZBA
      AYE  = CFA12*AXBA+CFA22*AYBA+CFA32*AZBA
      AZE  = CFA13*AXBA+CFA23*AYBA+CFA33*AZBA
C
C**INTEGRATE ACCELERATIONS
      VXED = AXE
      VYED = AYE
      VZED = AZE + AGRAV
C
C** CALCULATE TOTAL MISSILE ACCELERATION IN BODY AXES
      VDXB = CFA11*VXED + CFA12*VYED + CFA13*VZED
      VDYB = CFA21*VXED + CFA22*VYED + CFA23*VZED
      VDZB = CFA31*VXED + CFA32*VYED + CFA33*VZED
      ANGX = VDXB/32.174
      ANGY = VDYB/32.174
      ANGZ = VDZB/32.174
C
C**INTEGRATE VELOCITIES TO EARTH AXES  POSITION
   10 RXED = VXE
      RYED = VYE
      RZED = VZE
C
C**TARGET MOTION
      IF (OPTARG .LE. 0.) RETURN
      VTARGD = ATHRST*AGRAV
      BPSITD= 0.
      IF (VTARG.GT.0.) BPSITD= ATURNT*AGRAV*CRAD/VTARG
C
      VTXE = VTARG*COSD(BGAMT)*COSD(BPSIT)
      VTYE = VTARG*COSD(BGAMT)*SIND(BPSIT)
      VTZE = VTARG*SIND(BGAMT)
C
      RTXED = VTXE
      RTYED = VTYE
      RTZED = VTZE
C
      VDELX = VTXE-VXE
      VDELY = VTYE-VYE
```

```
      VDELZ = VTZE-VZE
C
      VCLSNG = (RDELX*VDELX+RDELY*VDELY+RDELZ*VDELZ)/RANGE
      RETURN
      END
```

```
      SUBROUTINE D1I
C**   TRANSLATIONAL DYNAMICS INITIALIZATION MODULE FOR D1
      COMMON C(3830)
      EQUIVALENCE (C(2561),N      )
      EQUIVALENCE (C(2562),IPL(1))
      DIMENSION IPL(100), ISNDX(40), ITNDX(10)
      EQUIVALENCE (C(3634), ISNDX(1)), (C(3512), I3512)
C
C** INPUT DATA
      EQUIVALENCE (C( 100),VWXE  )
      EQUIVALENCE (C( 101),VWYE  )
      EQUIVALENCE (C( 102),VWZE  )
      EQUIVALENCE (C( 204),VMACH )
      EQUIVALENCE (C( 367),BALPHA)
      EQUIVALENCE (C( 368),BALPHY)
      EQUIVALENCE (C( 427),BTHTG )
      EQUIVALENCE (C( 431),BPSIG )
      EQUIVALENCE (C(1639),OPTARG)
      EQUIVALENCE (C(1666),BLOSV )
      EQUIVALENCE (C(1667),RSLANT)
      EQUIVALENCE (C(1674),VMWTE )
      EQUIVALENCE (C(1751),CRAD  )
      EQUIVALENCE (C(3502),OPTN2 )
      EQUIVALENCE (C(3504),OPTN4 )
      EQUIVALENCE (C(3506),OPTN6 )
C
C** OUTPUT TO MODULES
      EQUIVALENCE (C(1615),RXE   )
      EQUIVALENCE (C(1619),RYE   )
      EQUIVALENCE (C(1623),RZE   )
      EQUIVALENCE (C(1603),VXE   )
      EQUIVALENCE (C(1607),VYE   )
      EQUIVALENCE (C(1611),VZE   )
      EQUIVALENCE (C(1651),RTXE  )
      EQUIVALENCE (C(1655),RTYE  )
      EQUIVALENCE (C(1659),RTZE  )
      EQUIVALENCE (C(1668),RXO   )
      EQUIVALENCE (C(1669),RYO   )
      EQUIVALENCE (C(1670),RZO   )
      EQUIVALENCE (C(1671),VXO   )
      EQUIVALENCE (C(1672),VYO   )
      EQUIVALENCE (C(1673),VZO   )
      EQUIVALENCE (C(1752),BPHIO )
      EQUIVALENCE (C(1753),BTHTO )
      EQUIVALENCE (C(1754),BPSIO )
      EQUIVALENCE (C(1665),RHZRO  )
      EQUIVALENCE (C(1635), RDELX)
      EQUIVALENCE (C(1636), RDELY)
      EQUIVALENCE (C(1637), RDELZ)
      EQUIVALENCE (C(1680),RSJYMC)
      EQUIVALENCE (C(1681),RSJZMC)
      EQUIVALENCE (C(3753),ITNDX(1)),(C(3721),ITCT)
      EQUIVALENCE (C(1761),  A011)
      EQUIVALENCE (C(1762),  A012)
      EQUIVALENCE (C(1763),  A013)
      EQUIVALENCE (C(1755),  A021)
      EQUIVALENCE (C(1756),  A022)
      EQUIVALENCE (C(1757),  A023)
      EQUIVALENCE (C(1758),  A031)
```

```
      EQUIVALENCE (C(1759),  A032)
      EQUIVALENCE (C(1760),  A033)
      EQUIVALENCE (C(1764),    P1)
      EQUIVALENCE (C(1765),    Q1)
      EQUIVALENCE (C(1766),    P2)
      EQUIVALENCE (C(1767),    R2)
      EQUIVALENCE (C(1768),  XB01)
      EQUIVALENCE (C(1769),  YB01)
      EQUIVALENCE (C(1770),  ZB01)
      EQUIVALENCE (C(1771),  XB02)
      EQUIVALENCE (C(1772),  YB02)
      EQUIVALENCE (C(1773),  ZB02)
      EQUIVALENCE (C( 360),BPH1ER)
      EQUIVALENCE (C( 361),BTH2ER)
      EQUIVALENCE (C( 362),BPS1ER)
      EQUIVALENCE (C(1562),GSPOTY)
      EQUIVALENCE (C(1572),GSPOTZ)
          EQUIVALENCE (C(1581),SIGPOT)
      EQUIVALENCE (C(1579),   ZETA)
      EQUIVALENCE (C(1580),     WO)
C
C
C* ZERO OUT SPOT JITTER MAX/MIN STORAGE LOCATIONS THAT ARE SAVED IN OUTP
      C(1567) = 0.
      C(1568) = 0.
      C(1577) = 0.
      C(1578) = 0.
C  PRINTED FROM MODULE G4
      WO = 3.94
      ZETA = .745
C
C SPOT JITTER MONTE CARLO INITIAL VALUES
C
      RSJYMC = 0.
      RSJZMC = 0.
      DO 500 IOL=1,ITCT
      ITSNDX = IOL
      IF(ITNDX(IOL).NE.1680) GO TO 502
      IPL(N)=1560
      IPL(N+1)=1563
      N=N+2
           IF(SIGPOT.NE.0.) GSPOTY
     1          = .707*SIGPOT/SQRT(WO/4./ZETA * C(2664))
      CALL MCARLO(RNSTRT,4,ITSNDX)
  502 IF(ITNDX(IOL).NE.1681) GO TO 500
      IPL(N)=1570
      IPL(N+1)=1573
      N=N+2
            IF(SIGPOT.NE.0.)
     1  GSPOTZ = .707*SIGPOT/SQRT(WO/4./ZETA * C(2664))
      CALL MCARLO(RNSTRT,4,ITSNDX)
  500 CONTINUE
C
      IPL(N) = 1600
      IPL(N+1) = 1604
      IPL(N+2) = 1608
      IPL(N+3) = 1612
      IPL(N+4) = 1616
      IPL(N+5) = 1620
      IPL(N+6) = 1640
```

```
      IPL(N+7) = 1644
      IPL(N+8) = 1648
      IPL(N+9) = 1652
      IPL(N+10) = 1656
      N = N+11
      C( 363) = 0.
      C( 364) = 0.
      CRAD = 57.295778
C
      IF (OPTN2.LE.0.) .GO TO 1002
      IF (OPTARG .LE. 0.) N = N-5
C
C**CALCULATE MISSILE PARAMETER INITIAL CONDITIONS
      RYE=0.
      RTZE = 0.
      RTYE = 0.
      RTXE = 0.
      BPHIO = 0.
      XB01 = 0.
      YB01 = 1.
      ZB01 = 0.
      XB02 = 0.
      YB02 = 0.
      ZB02 = 1.
C
C***  MONTE CARLO AUTOPILOT GYRO DRIFT RATES
      P1 = 0.
      Q1 = 0.
      P2 = 0.
      R2 = 0.
      DO 503 I = 1,I3512
      IDO = I
      IF(ISNDX(I).EQ.1764) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1765) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1766) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1767) CALL MCARLO (DUM, 1, IDO)
  503 CONTINUE
C
C
C*** AUTOPILOT GYRO BIAS ERRORS
C
1002  BPH1ER = 0.
      BTH2ER = 0.
      BPS1ER = 0.
      DO 11 I = 1, I3512
      IDO = I
      IF(ISNDX(I).EQ.360) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.361) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.362) CALL MCARLO (DUM, 1, IDO)
   11 CONTINUE
C
C** INITIALIZE MATRIX COEF FOR AUTOPILOT GYRO MODELS
C
      USPHI1 = SIND(BPHIO + BPH1ER)
      UCPHI1 = COSD(BPHIO + BPH1ER)
      USTHT2 = SIND(BTHTO + BTH2ER)
      UCTHT2 = COSD(BTHTO + BTH2ER)
      USPSI1 = SIND(BPSIO + BPS1ER)
      UCPSI1 = COSD(BPSIO + BPS1ER)
      A011 = UCPSI1*UCTHT2
```

```
      A012 = USPSI1*UCTHT2
      A013 = -USTHT2
      A021 = -USPSI1*UCPHI1 + UCPSI1*USTHT2*USPHI1
      A022 = UCPSI1*UCPHI1 + USPSI1*USTHT2*USPHI1
      A023 = UCTHT2*USPHI1
      A031 = UCPSI1*USTHT2*UCPHI1 + USPSI1*USPHI1
      A032 = USPSI1*USTHT2*UCPHI1 - UCPSI1*USPHI1
      A033 = UCTHT2*UCPHI1
      IF (OPTN2 .LE. 0) RETURN
C
C MISSILE INITIAL ATTITUDE ERRORS
C
      DO 5 I = 1, I3512
      IDO = I
      IF(ISNDX(I).EQ.1752) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1753) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.1754) CALL MCARLO (DUM, 1, IDO)
    5 CONTINUE
C
      IF (OPTN2.GT.1.0) GO TO 10
      RXE=-RSLANT*COSD(BLOSV)
      RZE=RSLANT*SIND(BLOSV)
      GO TO 20
   10 RSLANT = SQRT(RZE**2 + RXE**2)
   20 RH = RHZRO - RZE
C
      IF (OPTN4 .GT. 0.) GO TO 30
      BPSIO = CRAD*ARSIN(SIND(BPSIG)*RSLANT/RXE)
      CPSIO  = COSD(BPSIO)
      TTHTG  = SIND(BTHTG)/COSD(BTHTG)
      BTHTO = ATAND((-RZE/RXE - TTHTG*CPSIO),(CPSIO - TTHTG*RZE/RXE))
      GO TO 40
   30 CONTINUE
      IF (OPTN4 .GT. 1.) GO TO 40
      UST = SIND(BTHTO)
      USP = SIND(BPSIO)
      UCP = COSD(BPSIO)
      UCT = COSD(BTHTO)
      UCPH = COSD(BPHIO)
      USPH = SIND(BPHIO)
      RXBA = -RXE*UCP*UCT + RZE*UST
      RYBA = -RXE*(UCP*UST*USPH - USP*UCPH) - RZE*UCT*USPH
      RZBA = -RXE*(UCP*UST*UCPH + USP*USPH) - RZE*UCT*UCPH
      BTHTG = ATAND(-RZBA,RXBA)
      BPSIG = ATAND( RYBA,(RXBA*COSD(BTHTG)-RZBA*SIND(BTHTG)))
   40 CONTINUE
C
   24 VSOUND = 1117.3 - .00392*RH
      IF (OPTN6 .LE. 0.) VMWTE = VMACH*VSOUND
      VMWXY = VMWTE*COSD(BALPHA - BTHTO)
C
      VXE = VMWXY * COSD(BALPHY + BPSIO)
      VYE = VMWXY * SIND(BALPHY + BPSIO)
      VZE = VMWTE * SIND(BALPHA - BTHTO)
C
      RXO = RXE
      RYO = RYE
      RZO = RZE
      VXO = VXE
      VYO = VYE
```

```
VZO = VZE
RETURN
END
```

```
      SUBROUTINE D2
C**   ROTATIONAL DYNAMICS MODULE
      COMMON C(3830)
C
C**DATA INPUTS
      EQUIVALENCE (C(1748),FMIX  )
      EQUIVALENCE (C(1749),FMIY  )
      EQUIVALENCE (C(1750),FMIZ )
      EQUIVALENCE (C(1751),CRAD  )
      EQUIVALENCE (C(3503),OPTN3)
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(1303),FMXBA )
      EQUIVALENCE (C(1304),FMYBA )
      EQUIVALENCE (C(1305),FMZBA )
C
C**STATE VARIABLE OUTPUTS
      EQUIVALENCE (C(1700),CFA11D)
      EQUIVALENCE (C(1703),CFA11 )
      EQUIVALENCE (C(1704),CFA12D)
      EQUIVALENCE (C(1707),CFA12 )
      EQUIVALENCE (C(1708),CFA13D)
      EQUIVALENCE (C(1711),CFA13 )
      EQUIVALENCE (C(1712),CFA21D)
      EQUIVALENCE (C(1715),CFA21 )
      EQUIVALENCE (C(1716),CFA22D)
      EQUIVALENCE (C(1719),CFA22 )
      EQUIVALENCE (C(1720),CFA23D)
      EQUIVALENCE (C(1723),CFA23 )
      EQUIVALENCE (C(1724),CFA31D)
      EQUIVALENCE (C(1727),CFA31 )
      EQUIVALENCE (C(1728),CFA32D)
      EQUIVALENCE (C(1731),CFA32 )
      EQUIVALENCE (C(1732),CFA33D)
      EQUIVALENCE (C(1735),CFA33 )
      EQUIVALENCE (C(1736),WPD    )
      EQUIVALENCE (C(1739),WP     )
      EQUIVALENCE (C(1740),WQD    )
      EQUIVALENCE (C(1743),WQ     )
      EQUIVALENCE (C(1744),WRD    )
      EQUIVALENCE (C(1747),WR     )
C
C***** YAW PARAMETER INPUT
      EQUIVALENCE(C(2901),OPTNYW)
C
C**INTEGRATE BODY ANGULAR RATES
      IF (OPTN3.LE.0.) GO TO 45
      IF (OPTNYW.LE.0.) GO TO 55
      GO TO 65
   45 WPD = CRAD*FMXBA/FMIX
   55 WRD = (CRAD*FMZBA+(FMIX-FMIY)*WP*WQ/CRAD)/FMIZ
   65 WQD = (CRAD*FMYBA+(FMIZ-FMIX)*WP*WR/CRAD)/FMIY
C
C**INTEGRATE ATTITUDE DIRECTION COSINES
   49 CFA11D=(CFA21*WR-CFA31*WQ)/CRAD
      CFA12D=(CFA22*WR-CFA32*WQ)/CRAD
      CFA13D=(CFA23*WR-CFA33*WQ)/CRAD
      CFA21D = (CFA31*WP-CFA11*WR)/CRAD
      CFA22D = (CFA32*WP-CFA12*WR)/CRAD
```

```
CFA23D = (CFA33*WP-CFA13*WR)/CRAD
CFA31D = (CFA11*WQ-CFA21*WP)/CRAD
CFA32D = (CFA12*WQ-CFA22*WP)/CRAD
CFA33D = (CFA13*WQ-CFA23*WP)/CRAD
RETURN
END
```

```
      SUBROUTINE D2I
C**ROTATIONAL DYNAMICS INITIALIZATION MODULE D2IEUL
      COMMON C(3830)
      DIMENSION  IPL (100)
C**INPUT DATA
      EQUIVALENCE (C(1752),BPHIO )
      EQUIVALENCE (C(1753),BTHTO )
      EQUIVALENCE (C(1754),BPSIO )
C**INPUTS FROM MAIN PROGRAM
      EQUIVALENCE (C(2561),N      )
      EQUIVALENCE (C(2562),IPL    )
C**STATE VARIABLE OUTPUTS
      EQUIVALENCE (C(1703),CFA11 )
      EQUIVALENCE (C(1707),CFA12 )
      EQUIVALENCE (C(1711),CFA13 )
      EQUIVALENCE (C(1715),CFA21 )
      EQUIVALENCE (C(1719),CFA22 )
      EQUIVALENCE (C(1723),CFA23 )
      EQUIVALENCE (C(1727),CFA31 )
      EQUIVALENCE (C(1731),CFA32 )
      EQUIVALENCE (C(1735),CFA33 )
C**OTHER OUTPUTS
      EQUIVALENCE (C(1755),A021  )
      EQUIVALENCE (C(1756),A022  )
      EQUIVALENCE (C(1757),A023  )
      EQUIVALENCE (C(1758),A031  )
      EQUIVALENCE (C(1759),A032  )
      EQUIVALENCE (C(1760),A033  )
C**INITIAL CALCULATION OF EULER ANGLE MATRIX OF DIRECTION COSINES (CFA)
      USPHI = SIND(BPHIO)
      UCPHI = COSD(BPHIO)
      USTHT = SIND(BTHTO)
      UCTHT = COSD(BTHTO)
      USPSI = SIND(BPSIO)
      UCPSI = COSD(BPSIO)
      CFA11 = UCPSI*UCTHT
      CFA12 = USPSI*UCTHT
      CFA13 = -USTHT
      CFA21 = -USPSI*UCPHI+UCPSI*USTHT*USPHI
      CFA22 = UCPSI*UCPHI+USPSI*USTHT*USPHI
      CFA23 = UCTHT*USPHI
      CFA31 = UCPSI*USTHT*UCPHI+USPSI*USPHI
      CFA32 = USPSI*USTHT*UCPHI-UCPSI*USPHI
      CFA33 = UCTHT*UCPHI
C
C**INITIALIZE MATRIX COEF FOR FREE GYRO MODEL(S)
C
C**INTEGRATED PARAMATER LIST (IPL) FOR WPD,WQD,WRD,AND CFAD
      IPL(N) = 1700
      IPL(N+1) = 1704
      IPL(N+2) = 1708
      IPL(N+3) = 1712
      IPL(N+4) = 1716
      IPL(N+5) = 1720
      IPL(N+6) = 1724
      IPL(N+7) = 1728
      IPL(N+8) = 1732
      IPL(N+9) = 1736
      IPL(N+10) = 1740
```

```
IPL(N+11) = 1744
N = N+12
RETURN
END
```

```
      FUNCTION FINTP1(X,XI,YI,N,F,XL)
      DIMENSION XI(N), YI(N)
      IF(F .GT. 0.)GO TO 30
      DO 10 I=2, N
      IF(X .LE. XI(I)) GO TO 20
   10 CONTINUE
      I = N
   20 PCT = (X-XI(I-1))/(XI(I)-XI(I-1))
      F = 1.
   30 FINTP1 = YI(I-1) + PCT*(YI(I)-YI(I-1))
      RETURN
      END
```

```
      FUNCTION FINTP2(X,Y,XI,YI,ZI,NX,NY,NXY,F,XL)
      DIMENSION XI(NX),YI(NY), ZI(NXY), T(2), COL(10)
      IF(F .GT. 0.) GO TO 30
      DO 10 I=2, NY
      IF(Y .LE. YI(I)) GO TO 20
   10 CONTINUE
      I = NY
   20 PCT = (Y-YI(I-1))/(YI(I)-YI(I-1))
   30 DO 40 J=1,2
      L = I + J - 2
      L1NX = (L-1)*NX
      L1NX1 = L1NX + 1
      LNX = L * NX
      DO 50 IR = L1NX1, LNX
   50 COL(IR-L1NX) = ZI(IR)
   40 T(J) = FINTP1(X,XI,COL,NX,F,XL)
      FINTP2 = T(1) + PCT*(T(2)-T(1))
      RETURN
      END
```

```
      SUBROUTINE G3
C**AIR DATA MODULE G3
      COMMON C(3830)
C**INPUT DATA
      EQUIVALENCE (C(0208),RHZRO )
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(0100),VWXE  )
      EQUIVALENCE (C(0101),VWYE  )
      EQUIVALENCE (C(0102),VWZE  )
      EQUIVALENCE (C(1603),VXE   )
      EQUIVALENCE (C(1607),VYE   )
      EQUIVALENCE (C(1611),VZE   )
      EQUIVALENCE (C(1623),RZE  )
C**INPUTS FROM MAIN PROGRAM
C**STATE VARIABLE OUTPUTS
C**NONE
C**OTHER OUTPUTS
      EQUIVALENCE (C(0200),VMWXE )
      EQUIVALENCE (C(0201),VMWYE )
      EQUIVALENCE (C(0202),VMWZE )
      EQUIVALENCE (C(0203),PDYNMC)
      EQUIVALENCE (C(0204),VMACH )
      EQUIVALENCE (C(0205),DRHO  )
      EQUIVALENCE (C(0206),VSOUND)
      EQUIVALENCE (C(0207),VAIRSP)
      EQUIVALENCE (C(0209),RH    )
C**CALCULATE PRESENT ALTITUDE
      RH= -RZE+RHZRO
C**CALCULATE MISSILE VELOCITY WRT AIR MASS IN EARTH AXES
      VMWXE = VXE-VWXE
      VMWYE = VYE-VWYE
      VMWZE = VZE-VWZE
      VAIRSP = SQRT(VMWXE*VMWXE+VMWYE*VMWYE+VMWZE*VMWZE)
C**AIR DENSITY, SPEED OF SOUND, DYNAMIC PRESSURE, AND MACH
      DRHO=(.076475)/(1.+.3325E-04*RH+RH*RH*RH*.02315E-12)
      VSOUND = -.00392*RH+1117.3
      PDYNMC = (DRHO*VAIRSP*VAIRSP)/64.344
      VMACH = VAIRSP/VSOUND
      RETURN
      END
```

```
      SUBROUTINE G4
C*****************************************************************
C** THIS IS A SUBROUTINE (NOT A MODULE) CALLED BY STAGE 3 **
C** STOPS PROGRAM AND COMPUTES MISS DISTANCE                     **
C*****************************************************************
      COMMON C(3830)
      COMMON /XMA/XMAX(4,7)
  100 FORMAT(1H0,17H MISS DISTANCE = ,1PE15.7/
     *       1H0,17H FLIGHT  TIME  = ,1PE15.7)
  200 FORMAT(1H0, 9X,8HRDELX = ,1PE15.7, 8X,8HRDELY = ,1PE15.7,
     *            8X,8HRDELZ = ,1PE15.7)
  300 FORMAT(1H0,40X,8HRYFP  = ,1PE15.7, 8X,8HRZFP  = ,1PE15.7)
      EQUIVALENCE (C( 357),BGAMH )
     *,           (C( 358),BGAMV )
     *,           (C( 371),RANGE )
     *,           (C(1635),RDELX )
     *,           (C(1636),RDELY )
     *,           (C(1637),RDELZ )
      EQUIVALENCE (C(2000),T     )
      EQUIVALENCE (C(1564),YMC)
      EQUIVALENCE (C(1565),YMC2)
      EQUIVALENCE (C(1574),ZMC)
      EQUIVALENCE (C(1575),ZMC2)
     *,           (C(2020),LCONV )
      EQUIVALENCE (C( 300),RMISS )
     $,           (C(301),L      )
     *,           (C( 302),RYF   )
     *,           (C( 303),RZF   )
      EQUIVALENCE (C( 31), LCEP)
      EQUIVALENCE (C(3721),   ITCT)
      EQUIVALENCE (C(3000),VSD   (1))
      EQUIVALENCE (C(3010),VMEAN (1))
      EQUIVALENCE (C(3020),IMVNDX(1))
      EQUIVALENCE (C(3030),IMVCT )
      EQUIVALENCE (C(1651),RTXE),(C(1655),RTYE), (C(1659),RTZE),(C(1615)
     .,RXE), (C(1619),RYE),(C(1623),RZE)
      REAL*8 PITCH, YAW
      DATA PITCH,YAW/8HPITCH   ,8HYAW     /
      DIMENSION IMVNDX(10), VMEAN(10), VSD(10)
      DIMENSION VSUM(10), VS2(10)
      LCEP = 0
      RDELX=RTXE-RXE
      RDELY=RTYE-RYE
      RDELZ=RTZE-RZE
      IF(RDELZ .LT. 0. .OR. RDELX .LT. 0.) LCONV=2
      IF (RANGE .GT. 500.) GO TO 20
      UC13 =-SIND(BGAMV)
      UC33 = COSD(BGAMV)
      UC21 =-SIND(BGAMH)
      UC22 = COSD(BGAMH)
      UC11 =  UC22*UC33
      UC12 = -UC21*UC33
      UC31 = -UC22*UC13
      UC32 =  UC21*UC13
      RXFP = UC11*RDELX + UC12*RDELY + UC13*RDELZ
      RYFP = UC21*RDELX + UC22*RDELY
      RZFP = UC31*RDELX + UC32*RDELY + UC33*RDELZ
      IF (RXFP .GT. 0.) GO TO 10
      IF(IMVCT .LE. 0) GO TO 50
```

```
      DO 5 I=1,IMVCT
      IDO = IMVNDX(I)
      VALUE = C(IDO)
      VSUM(I) = VSUM(I) + VALUE
      VS2(I) = VS2(I) + VALUE**2
      TCASE = L
      TCASE1 = L - 1
      VMEAN(I) = VSUM(I)/TCASE
      IF(L .NE. 1) GO TO 2
      S2 = (VS2(I) - (VSUM(I)**2)/TCASE)/TCASE
      GO TO 3
    2 S2 = (VS2(I) - (VSUM(I)**2)/TCASE)/TCASE1
    3 VSD(I) = SQRT(S2)
    5 CONTINUE
   50 CONTINUE
      PCT = UXFP/(RXFP - UXFP)
      RDX = UDELX - PCT*(RDELX - UDELX)
      RDY = UDELY - PCT*(RDELY - UDELY)
      RDZ = UDELZ - PCT*(RDELZ - UDELZ)
      RYF = UYFP - PCT*(RYFP - UYFP)
      RZF = UZFP - PCT*(RZFP - UZFP)
      TZERO = UT - PCT*(T - UT)
      RMISS = SQRT(RYF**2 + RZF**2)
      WRITE(6,600)C(630),PITCH
      WRITE(6,600)C(631),YAW
600   FORMAT(1H0,60X,24H+++MAX BREAKLOCK VALUE =F10.5,5H  IN ,A8)
      WRITE(6,400) L
  400 FORMAT(1H0,13HRUN NUMBER = ,I2)
      IF(ITCT.LE.0)GO TO 30
      CALL MCARLX(DUM,2,RNSTRT)
      WRITE(6,500) C(1567), C(1568), C(1577), C(1578)
      XMCSPT = SQRT(YMC2*YMC2 + ZMC2*ZMC2)
      WRITE(6,2555)YMC,YMC2
      WRITE(6,2556)ZMC,ZMC2,XMCSPT
30    CONTINUE
  500 FORMAT(1H0,11X,13HMAX SPOT Y = ,F6.2,14H MIN SPOT Y = , F6.2/
     1             12X,13HMAX SPOT Z = ,F6.2,14H MIN SPOT Z = ,F6.2//
     2 )
 2555 FORMAT(1H0,11X,26HSAMPLE SPOT JITTER Y-MEAN=,F10.5,6X,12HMEAN SQUA
     1RE=,F10.5)
 2556 FORMAT(1H0,11X,26HSAMPLE SPOT JITTER Z-MEAN=,F10.5,6X,12HMEAN SQUA
     1RE=,F10.5,6X,18HSPOT RADIAL RMS = ,F10.5)
      WRITE(6,100) RMISS, TZERO
      WRITE(6,17)XMAX
 17    FORMAT(7E17.8)
      WRITE(6,200) RDX, RDY, RDZ
      WRITE(6,300) RYF, RZF
      LCONV = 2
      LCEP = 1
      RETURN
   10 UT = T
      UDELX = RDELX
      UDELY = RDELY
      UDELZ = RDELZ
      UXFP = RXFP
      UYFP = RYFP
      UZFP = RZFP
      RETURN
   20 CONTINUE
      RETURN
```

```
      SUBROUTINE G5
C**COORDINATE CONVERSION MODULE
      COMMON C(3830)
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(0200),VMWXE )
      EQUIVALENCE (C(0201),VMWYE )
      EQUIVALENCE (C(0202),VMWZE )
      EQUIVALENCE (C(0207),VAIRSP)
      EQUIVALENCE (C(1317),RAIL  )
      EQUIVALENCE (C(1405),QBURN )
      EQUIVALENCE (C(1603),VXE   )
      EQUIVALENCE (C(1607),VYE   )
      EQUIVALENCE (C(1611),VZE   )
      EQUIVALENCE (C(1615),RXE   )
      EQUIVALENCE (C(1619),RYE   )
      EQUIVALENCE (C(1623),RZE   )
      EQUIVALENCE (C(1635),RDELX )
      EQUIVALENCE (C(1636),RDELY )
      EQUIVALENCE (C(1637),RDELZ )
      EQUIVALENCE (C(1651),RTXE  )
      EQUIVALENCE (C(1655),RTYE  )
      EQUIVALENCE (C(1659),RTZE  )
      EQUIVALENCE (C(1668),RXO   )
      EQUIVALENCE (C(1669),RYO   )
      EQUIVALENCE (C(1670),RZO   )
      EQUIVALENCE (C(1671),VXO   )
      EQUIVALENCE (C(1672),VYO   )
      EQUIVALENCE (C(1673),VZO   )
      EQUIVALENCE (C(1680),RSJYMC)
      EQUIVALENCE (C(1681),RSJZMC)
      EQUIVALENCE (C(1682),RSPOTX)
      EQUIVALENCE (C(1683),RSPOTY)
      EQUIVALENCE (C(1684),RSPOTZ)
      EQUIVALENCE (C(3753), ITNDX(1))
      EQUIVALENCE (C(3721),  ITCT)
      EQUIVALENCE (C(1560), SXPDD)
      EQUIVALENCE (C(1561),    RX)
      EQUIVALENCE (C(1562),GSPOTY)
      EQUIVALENCE (C(1563),  SXPD)
      EQUIVALENCE (C(1566),   SXP)
      EQUIVALENCE (C(1570), SYPDD)
      EQUIVALENCE (C(1571),    RY)
      EQUIVALENCE (C(1572),GSPOTZ)
      EQUIVALENCE (C(1573),  SYPD)
      EQUIVALENCE (C(1576),   SYP)
      EQUIVALENCE (C(1579),   ZETA)
      EQUIVALENCE (C(1580),     WO)
      DIMENSION ITNDX(10)
      EQUIVALENCE (C(1703),CFA11 )
      EQUIVALENCE (C(1707),CFA12 )
      EQUIVALENCE (C(1711),CFA13 )
      EQUIVALENCE (C(1715),CFA21 )
      EQUIVALENCE (C(1719),CFA22 )
      EQUIVALENCE (C(1723),CFA23 )
      EQUIVALENCE (C(1727),CFA31 )
      EQUIVALENCE (C(1731),CFA32 )
      EQUIVALENCE (C(1735),CFA33 )
      EQUIVALENCE (C(1751),CRAD  )
```

```
      EQUIVALENCE (C(1768),  XB01)
      EQUIVALENCE (C(1769),  YB01)
      EQUIVALENCE (C(1770),  ZB01)
      EQUIVALENCE (C(1771),  XB02)
      EQUIVALENCE (C(1772),  YB02)
      EQUIVALENCE (C(1773),  ZB02)
      EQUIVALENCE (C(1764),    P1)
      EQUIVALENCE (C(1765),    Q1)
      EQUIVALENCE (C(1766),    P2)
      EQUIVALENCE (C(1767),    R2)
      EQUIVALENCE (C(1761),  A011)
      EQUIVALENCE (C(1762),  A012)
      EQUIVALENCE (C(1763),  A013)
      EQUIVALENCE (C(1755),A021  )
      EQUIVALENCE (C(1756),A022  )
      EQUIVALENCE (C(1757),A023  )
      EQUIVALENCE (C(1758),A031  )
      EQUIVALENCE (C(1759),A032  )
      EQUIVALENCE (C(1760),A033  )
      EQUIVALENCE (C(2000),T     )
C
C**OTHER OUTPUTS
      EQUIVALENCE (C(0350),BTHT  )
      EQUIVALENCE (C(0351),BPSI  )
      EQUIVALENCE (C(0352),BPHI  )
      EQUIVALENCE (C( 353),BPH1  )
      EQUIVALENCE (C( 354),BTH2  )
      EQUIVALENCE (C( 355),BPS1  )
      EQUIVALENCE (C(0356),VTOTE )
      EQUIVALENCE (C(0357),BGAMH )
      EQUIVALENCE (C(0358),BGAMV )
      EQUIVALENCE (C(0363),BTHLV )
      EQUIVALENCE (C(0364),BPSLV )
      EQUIVALENCE (C(0365),BLAMV )
      EQUIVALENCE (C(0366),BLAMH )
      EQUIVALENCE (C(0367),BALPHA)
      EQUIVALENCE (C(0368),BALPHY)
      EQUIVALENCE (C(0369),BALPHP)
      EQUIVALENCE (C(0370),BPHIP )
      EQUIVALENCE (C(0371),RANGE )
      EQUIVALENCE (C(0372),RXBA  )
      EQUIVALENCE (C(0373),RYBA  )
      EQUIVALENCE (C(0374),RZBA  )
      EQUIVALENCE (C(1663),VDXB  )
      EQUIVALENCE (C(1664),VDYB  )
      EQUIVALENCE (C(1665),VDZB  )
      EQUIVALENCE (C( 377),BALPD )
      EQUIVALENCE (C( 378),BALYD )
      EQUIVALENCE (C( 380),RANGO )
      EQUIVALENCE (C( 390),RXL   )
      EQUIVALENCE (C( 391),RYL   )
      EQUIVALENCE (C( 392),RZL   )
      EQUIVALENCE (C( 393),BPH2  )
C
C**CALCULATION OF HEADING, PITCH, ROLL EULER ANGLES IN DEGREES
      BPHI = ATAND(CFA23,CFA33)
      BTHT = ATAND(-CFA13,SQRT(CFA11*CFA11+CFA12*CFA12))
      BPSI = ATAND(CFA12,CFA11)
C
C**FREE GYRO MODELS (INITIAL GIMBAL ANGLES ARE ZERO)
```

```
C
C** AUTO PILOT DRIFT RATES
      DXB01 = -Q1*YB01/CRAD
      DYB01 = (Q1*XB01 - P1*ZB01)/CRAD
      DZB01 = P1*YB01/CRAD
      DXB02 =  R2*ZB02/CRAD
      DYB02 = -P2*ZB02/CRAD
      DZB02 = (P2*YB02 - R2*XB02)/CRAD
C
      XB01 = DXB01*T
      YB01 = 1. + DYB01*T
      ZB01 = DZB01*T
      XB02 = DXB02*T
      YB02 = DYB02*T
      ZB02 = 1. + DZB02*T
      B11 = A011*CFA11 + A012*CFA12 + A013*CFA13
      B12 = A011*CFA21 + A012*CFA22 + A013*CFA23
      B13 = A011*CFA31 + A012*CFA32 + A013*CFA33
      B21 = A021*CFA11 + A022*CFA12 + A023*CFA13
      B22 = A021*CFA21 + A022*CFA22 + A023*CFA23
      B23 = A021*CFA31 + A022*CFA32 + A023*CFA33
      B31 = A031*CFA11 + A032*CFA12 + A033*CFA13
      B32 = A031*CFA21 + A032*CFA22 + A033*CFA23
      B33 = A031*CFA31 + A032*CFA32 + A033*CFA33
      XB1 = B11*XB01 + B21*YB01 + B31*ZB01
      YB1 = B12*XB01 + B22*YB01 + B32*ZB01
      ZB1 = B13*XB01 + B23*YB01 + B33*ZB01
      XB2 = B11*XB02 + B21*YB02 + B31*ZB02
      YB2 = B12*XB02 + B22*YB02 + B32*ZB02
      ZB2 = B13*XB02 + B23*YB02 + B33*ZB02
      BPH1 = ATAND (ZB1,YB1)
      BPS1 = ATAND (-XB1,YB1/COSD(BPH1))
      BTH2 = ATAND (XB2,ZB2)
      BPH2 = ATAND (-YB2,ZB2/COSD(BTH2))
C
C**CALCULATION OF TOTAL VELOCITY
      VTOTE = SQRT(VXE*VXE+VYE*VYE+VZE*VZE)
      RDELX = RTXE-RXE
      RDELY = RTYE-RYE
      RDELZ = RTZE-RZE
C
      IF(C(1976).LE.0.) GO TO 20
      RXL = RXE - RXO - VXO*T
      RYL = RYE - RYO - VYO*T
      RZL = RZE - RZO - VZO*T
      RANGO = SQRT(RXL**2 + RYL**2 + RZL**2)
      VXL = VXE - VXO
      VYL = VYE - VYO
      VZL = VZE - VZO
   20 CONTINUE
C
C**TRANSFORM MISSILE LOS FROM EARTH TO BODY AXES
C
C LINE OF SIGHT OF LASER SPOT WITH MONTE CARLO SPOT JITTER INCLUDED
C
      DO 500 I = 1, ITCT
      IDO = I
      IF(ITNDX(I).NE.1680) GO TO 501
      RSJYMC = GSPOTY*SXP
      CALL MCARLO (DUM,2,IDO)
```

```
      SXPDD = WO*WO*(RX-2.*ZETA*SXPD/WO - SXP)
  501 IF(ITNDX(I).NE.1681) GO TO 500
      RSJZMC = GSPOTZ*SYP
      CALL MCARLO (DUM,2,IDO)
      SYPDD = WO*WO*(RY - 2.*ZETA*SYPD/WO - SYP)
  500 CONTINUE
      RSPOTX = RDELX
      RSPOTY = RDELY + RSJYMC
      RSPOTZ = RDELZ + RSJZMC
      RXBA = RSPOTX*CFA11 + RSPOTY*CFA12 + RSPOTZ*CFA13
      RYBA = RSPOTX*CFA21 + RSPOTY*CFA22 + RSPOTZ*CFA23
      RZBA = RSPOTX*CFA31 + RSPOTY*CFA32 + RSPOTZ*CFA33
C
      UVP1 =  VXE*RDELX+VYE*RDELY
      UVP2 = RDELX*RDELX+RDELY*RDELY
      UVP3 = VZE*RDELZ
      UVP4 = SQRT(UVP2)
      RANGE = SQRT(UVP2+RDELZ**2)
C**VERTICAL AND HORIZONTAL LINE OF SIGHT ANGLES (EARTH AXES)
C
      BLAMH = ATAND(-RDELY,RDELX)
      BLAMV = ATAND(-RDELZ,UVP4)
C
C**VERTICAL AND HORIZONTAL PROPORTIONAL NAVIGATION ANGLES
      IF(VTOTE.LE.10.) GO TO 30
      VXP=(UVP1+UVP3)/RANGE
      VYP = ( VYE*RDELX-VXE*RDELY)/UVP4
      VZP = (VZE*UVP2-RDELZ*UVP1)/(RANGE*UVP4)
      BTHLV  = ATAND(VZP,VXP)
      BPSLV  = ATAND(VYP,VXP)
C
      BGAMV = ATAND(-VZE,SQRT(VXE*VXE+VYE*VYE))
      BGAMH = ATAND(VYE,VXE)
C
C**VELOCITY WRT AIR IN BODY AXES
      VMWU = CFA11*VMWXE+CFA12*VMWYE+CFA13*VMWZE
      VMWV = CFA21*VMWXE+CFA22*VMWYE+CFA23*VMWZE
      VMWW = CFA31*VMWXE+CFA32*VMWYE+CFA33*VMWZE
C
C**VERTICAL AND HORIZONTAL ANGLES OF ATTACK
      IF (QBURN.LE.0. .AND. RANGO.LE.RAIL) GO TO 30
      BALPHA = ATAND(VMWW,VMWU)
      BALPHY = ATAND(VMWV,VMWU)
      USQ=VMWU**2
      BALPD=(VMWU*VDZB-VMWW*VDXB)/(USQ+VMWW**2)*CRAD
      BALYD=(VMWU*VDYB-VMWV*VDXB)/(USQ+VMWV**2)*CRAD
C
C**ALPHA PRIME AND PHI PRIME (WIND TUNNEL AXES)
      IF ((BALPHA-BALPHY).EQ.0.) GO TO 30
      BPHIP = ATAND(BALPHY,BALPHA)
   30 BALPHP=SQRT(BALPHA**2+BALPHY**2)
      RETURN
      END
```

```
      SUBROUTINE LTRAN(T,DELT,AMP,Y,YC,IFLG,K)
      DIMENSION A(5,3), PHI(5,3), W(5,3)
      DATA IMAX,AE/4,-1./
      DATA A/1.,4.,12.,26.,0.,1.,4.,12.,26.,0.,1.,4.,12.,26.,0./
      IF(IFLG.GT.0)GO TO 17
      ZC=0.
      W1=6.28*11.
      DO 1 I=1,IMAX
      CALL RANNUM(0.,RNSTRT,RN)
      PHI(I,K)=3.14*RN
      W(I,K)=I*W1
C ZC IS INTEGRATION CONSTANT FOR Z
      B=W(I,K)*T+PHI(I,K)
      ZC=ZC+ A(I,K)*(AE*SIN(B)-W(I,K)*COS(B))/(AE**2+W(I,K)**2)
1     CONTINUE
      YC=AMP*EXP(AE*T)*ZC
17    CONTINUE
      Z=0.
      DO 2 I=1,IMAX
      Z=Z+A(I,K)*SIN(W(I,K)*T+PHI(I,K))
2     CONTINUE
      Y=AMP*EXP(AE*T)*Z
      RETURN
      END
```

```
      SUBROUTINE OINPT1
C     BASIC INPUT SUBROUTINE OINPT1
      COMMON C(3830)
      COMMON/WKU1/ONAME0(50)
      EQUIVALENCE (C(3218),ONAME1(1)), (C(3268),ONAME2(1)), (C(3318),ONA
     C      ME3(1)),
     C            (C(3328),ONAME4(1)), (C(2361),NOMOD ), (C(2362),MODNO
     C       (1)),
     C            (C(3168),OUTNO(1)), (C(2461),NOSUB ), (C(2462),SUBNO
     C      (1)),
     C      (C(3066),NOLIST),(C(3167),NOOUT),
     C      (C(3067),LISTNO(1)), (C(3117),VALUE(1) ), (C(2008),PLOTNO),
     C      (C(2009),NOPLOT), (C(2325),VLABLE(1,1)), (C(1),K(1))
      EQUIVALENCE (C(2010),  STEP)
      EQUIVALENCE (C(1984),NPLOT )
      EQUIVALENCE (C(1985),OUTPLT(1))
      EQUIVALENCE (C(3512),ISGCT),(C(3514),SIGMA(1)),(C(3554),SIGLB
     C      (1)),(C(3594),SIGUB(1)),
     * (C(3634),ISNDX(1)),(C(3674),IDIST(1)),(C(3511),RNSTRT)
      EQUIVALENCE(C(3721),ITCT),(C(3723),TSGMA(1)),(C(3733),TLB(1)),
     *(C(3743),TUB(1)),(C(3753),ITNDX(1)),(C(3763),ITDIST(1)),(C(3773),
     C      TSPER(1)),
     *(C(3783),TYPPER(1)),(C(3793),TPSIG(1)),(C(3803),TNXST(1)),(C(3813)
     C      ,ITNDX2(1))
      EQUIVALENCE (C( 21),IBVNSW)
      EQUIVALENCE (C( 22), IPLOT)
      EQUIVALENCE (C(19),PSIZE)
      EQUIVALENCE (C( 23),XLAMBD)
      EQUIVALENCE (C( 24), KSSIG)
      EQUIVALENCE (C( 25),CEPSIG(1))
      EQUIVALENCE (C(3825),  NCASE)
      EQUIVALENCE (C(3020),IMVNDX(1))
      EQUIVALENCE (C(3030),IMVCT )
      REAL*8 CPERTY,SSS
      DIMENSION ONAME3(10),ONAME4(10)
      DIMENSION LISTNO(50), VALUE(50)
      DIMENSION SUBNO(99),IR(4),VR(4)
      DIMENSION ALPHA(4),ONAME1(50),ONAME2(50),OUTNO(50) ,MODNO(99)
      DIMENSION K(3510)
      DIMENSION VLABLE(2,15)
      DIMENSION OUTPLT(15)
      DIMENSION SIGMA(40),SIGLB(40),SIGUB(40),ISNDX(40),IDIST(40)
      DIMENSION TSGMA(10),TLB(10),TUB(10),ITNDX(10),ITDIST(10),
     *TSPER(10),TYPPER(10),TPSIG(10),ITNDX2(10),TNXST(10)
      DIMENSION IMVNDX(10)
      DIMENSION CEPSIG(6)
      INTEGER CEPSIG
      REAL KSSIG
      REAL MODNO
      INTEGER OUTNO
      INTEGER OUTPLT
      DIMENSION COEFF(28),REF(28),BREF(28),ATD(5),BATD(5),DTA(4),BDTA(4)
      EQUIVALENCE (C(3359),COEFF(1))
      EQUIVALENCE (C(3387),REF(1))
      EQUIVALENCE (C(3415),BREF(1))
      EQUIVALENCE (C(3443),ATD(1))
      EQUIVALENCE (C(3448),BATD(1))
      EQUIVALENCE (C(3453),DTA(1))
      EQUIVALENCE (C(3457),BDTA(1))
```

```
      NAMELIST /DAP/COEFF,REF,BREF,ATD,BATD,DTA,BDTA
      DATA CPERTY/8HR       /
      DATA SSS/8HS       /
      JAR = 0
      WRITE(6,31)
   31 FORMAT(11H1INPUT DATA/)
    1 READ(5,2,END=50)IR(1),(ALPHA(JC),JC=1,4),IR(2),IR(3),TPER,TPSGMA,
     *VR(1),VR(2),VR(3),IR(4),VR(4)
   55 CONTINUE
      WRITE(6,30)IR(1),(ALPHA(JC),JC=1,4),IR(2),IR(3),TPER,TPSGMA,
     *VR(1),VR(2),VR(3),
     *IR(4),VR(4)
   30 FORMAT(1X,I2,4A4,I5,I2,A1,F5.2,2E15.7,F10.4,I5,F7.4)
    2 FORMAT(I2,4A4,I5,I1,A1,F3.2,2E15.7,F10.4,I5,F5.2)
    7 IF( IR(1) .NE. 1 )  GO TO 3
      NOSUB = NOSUB + 1
      SUBNO(NOSUB) = IR(2)
      GO TO 1
    3 IF( IR(1) .NE. 2 )  GO TO 4
      NOMOD = NOMOD + 1
      MODNO(NOMOD) = IR(2)
      GO TO 1
    4 IF(IR(1) .NE. 3) GO TO 5
      L = IR(2)
      C(L) = VR(1)
      IF (VR(2) .EQ. 0.) GO TO 1
      NOLIST = NOLIST + 1
      LISTNO(NOLIST) = L
      VALUE(NOLIST) = VR(1)
      GO TO 1
    5 IF(IR(1) .NE. 4)GO TO 6
      NOOUT = NOOUT + 1
      IF (NOOUT.GT.50) GO TO 1
      ONAME0(NOOUT)=ALPHA(1)
      ONAME1(NOOUT)=ALPHA(2)
      ONAME2(NOOUT) = ALPHA(3)
      OUTNO(NOOUT) = IR(2)
      GO TO 1
    6 IF (IR(1) .NE. 5) GO TO 16
      READ(5,DAP)
      WRITE(6,DAP)
      GO TO 1
   16 IF (IR(1).NE.7) GO TO 19
      NPLOT=NPLOT+1
      IF (NPLOT.GT.15) GO TO 1
      DO 20 I=1,2
   20 VLABLE (I,NPLOT)=ALPHA(I+1)
      OUTPLT(NPLOT)=IR(2)
      GO TO 1
   19 IF(IR(1).NE.8) GO TO 18
      IF(TPER.EQ.SSS) GO TO 194
      IF(VR(4).GT.0.) GO TO 192
      IF(IR(3).NE.0.AND.IR(3).NE.1) GO TO 193
      ISGCT=ISGCT+1
      SIGMA(ISGCT)=VR(1)
      SIGLB(ISGCT)=VR(2)
      SIGUB(ISGCT)=VR(3)
      ISNDX(ISGCT)=IR(2)
      IDIST(ISGCT)=IR(3)
      GO TO 1
```

```
   18 IF(IR(1) .NE. 9) GO TO 100
      STEP = 11.
      READ(5,8)NP,IBVNSW,IPLOT,XLAMBD,KSSIG,(CEPSIG(I),I=1,5),PSIZE
    8 FORMAT(3I4,2F10.3,5I2,E15.7)
      GO TO 1
  100 IF(IR(1) .NE. 10) GO TO 191
      IMVCT = IMVCT + 1
      IMVNDX(IMVCT) = IR(2)
      GO TO 1
  192 IF(IR(3).GT.5) GO TO 193
      ITCT=ITCT+1
      TSGMA(ITCT)=VR(1)
      TLB(ITCT)=VR(2)
      TUB(ITCT)=VR(3)
      ITNDX2(ITCT)=IR(2)
      IF(IR(4).GT.0) ITNDX2(ITCT)=IR(4)
      ITNDX(ITCT)=IR(2)
      ITDIST(ITCT)=IR(3)
      TSPER(ITCT)=VR(4)
      TPSIG(ITCT)=TPSGMA
      TYPPER(ITCT)=0.
      IF(TPER.EQ.CPERTY) TYPPER(ITCT)=1.
      GO TO 1
  194 RNSTRT=VR(1)
      GO TO 1
  193 WRITE(6,5518)
 5518 FORMAT(1X,58HUNDEFINED DISTRIBUTION TYPE NUMBER ENTERED - CARD REJ
     *ECTED)
      WRITE(6,30)IR(1),(ALPHA(JC),JC=1,4),IR(2),IR(3),TPER,TPSGMA,
     *VR(1),VR(2),VR(3),
     *IR(4),VR(4)
      GO TO 1
191   CONTINUE
      NCASE=NCASE+1
      RETURN
   50 STOP
      END
```

```
      SUBROUTINE OUPT2
C     OUTPUT INITIALIZITION SUBROUTINE OUPT2
      COMMON C(3830),GRAPH
      EQUIVALENCE (C(2017),DTCNT ), (C(3167),NOOUT ), (C(2016),PGCNT ),
     C            (C(2014),ITCNT ), (C(2003),PCNT  ), (C(2015),CPP   ),
     C            (C(2018),TAPE  ), (C(2019),TAPEND), (C(2013),DOC   ),
     C            (C(2000),T     ), (C(2021),KCONV ), (C(2025),TIME(1)),
     C            (C(2008),PLOTNO),(C(2009),NOPLOT),(C(3168),OUTNO(1)),
     C            (C(2004),PPNT  ), (C(2023),OPOINT)
      DIMENSION GRAPH(1,1),TIME(300),OUTNO(50)
      INTEGER     PGCNT , DTCNT , OUTNO , OPOINT
      EQUIVALENCE (C(1985),OUTPLT(1))
      INTEGER OUTPLT
      DIMENSION OUTPLT(15)
      KCONV=0
      ITCNT = DOC + 1.0
      PCNT = T-0.000001
      PGCNT = 1
      DTCNT = (NOOUT + 4)/5
      IF ( ITCNT .GE. 7) GO TO 2
      ITCNT = ITCNT + 1
      CALL DUMPO
C
    2 TIME(1)=T
      OPOINT =1
      IF(NOPLOT.EQ.0)GOTO 11
      DO 10 J=1,NOPLOT
      K=OUTPLT(J)
   10 GRAPH(1,J)=C(K)
   11 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OUPT3
C     OUTPUT SUBROUTINE OUPT3
      COMMON C(3830),GRAPH(300,4)
      COMMON/WKU1/ONAMEO(50)
      EQUIVALENCE (C(3168),OUTNO(1)), (C(3218),ONAME1(1)),
     C  (C(3268),ONAME2(1)),
     C            (C(2017),DTCNT ), (C(3167),NOOUT ), (C(2016),PGCNT ),
     C            (C(2014),ITCNT ), (C(2003),PCNT  ), (C(2015),CPP   ),
     C            (C(2000),T     ), (C(2664),DER   ), (C(2018),TAPE  ),
     C            (C(2019),TAPEND), (C(2008),PLOTNO), (C(2009),NOPLOT),
     C            (C(2005),PPP   ),(C(2004),PPNT  ),(C(2025),TIME(1)),
     C            (C(2023),OPOINT)
      EQUIVALENCE (C(1985),OUTPLT(1))
      DIMENSION B(50),OUTNO(50),ONAME1(50),ONAME2(50)
      DIMENSION TIME(300)
      DIMENSION OUTPLT(15)
      INTEGER DTCNT,PGCNT,OUTNO
      INTEGER OPOINT
      INTEGER OUTPLT
      DATA DER1/0.0/
C
C** SAVE SPOT JITTER MAX/MIN VALUES
      IF(C(1680).GT.C(1567)) C(1567) = C(1680)
      IF(C(1680).LT.C(1568)) C(1568) = C(1680)
      IF(C(1681).GT.C(1577)) C(1577) = C(1681)
      IF(C(1681).LT.C(1578)) C(1578) = C(1681)
C
      IF (ITCNT. GT. 6) GO TO 7
      ITCNT = ITCNT + 1
      CALL DUMPO
      PGCNT = 1
C
    7 IF (DER. EQ. DER1) GO TO 8
      DER1 = DER
      WRITE(6,20)T,DER
   20 FORMAT(1H ,5HTIME=, F14.7,2X,10HSTEP SIZE=,1PE19.7)
    8 IF (T .LT. PCNT)GOTO15
    9 PCNT = PCNT + CPP
      IF (PGCNT. NE. 1) GO TO 3
      IF(NOOUT.LE.1) GO TO 3
    1 WRITE(6,2) (ONAMEO(I),ONAME1(I),ONAME2(I), I=1,NOOUT)
    2 FORMAT (1H1,3X,4HTIME,5X,5(7X,3A4)//(20X,3A4,7X,3A4,7X,3A4,7X,
     13A4,7X,3A4)/)
      PGCNT = 2*DTCNT + 4
    3 IF(PGCNT .GE. 86) GO TO 1
      DO 4 I = 1,NOOUT
      J = OUTNO(I)
    4 B(I) = C(J)
      IF(NOOUT.LE.1) GO TO 15
      WRITE (6,5) T,(B(I), I = 1,NOOUT)
    5 FORMAT (///,F14.7,1P5E19.7/(14X,1P5E19.7))
      PGCNT = PGCNT + DTCNT + 4
   15 IF(T.LT.PPNT.OR.NOPLOT.EQ.0)RETURN
      PPNT=PPNT+PPP
      KPOINT =OPOINT +1
      IF (KPOINT-300) 16,13,18
   13 WRITE (6,14)
   14 FORMAT (//71H ****  WARNING-PLOTTING ARRAY FILLED-ONLY FIRST 300 P
     COINTS PLOTTED ****,//)
```

```
16 OPOINT=KPOINT
   TIME (OPOINT)=T
   IF(NOPLOT.EQ.0)GOTO 11
   DO 10 J=1,NOPLOT
   K=OUTPLT(J)
10 GRAPH(OPOINT   ,J)=C(K)
11 CONTINUE
18 RETURN
   END
```

```
SUBROUTINE STGE2
COMMON C(3830)
EQUIVALENCE (C(2011),KSTEP ), (C(2020),LCONV ), (C(2021),KCONV )
KCONV = 0
LCONV = 0
KSTEP = 1
RETURN
END
```

```
      SUBROUTINE STGE3
      COMMON C(3830)
      EQUIVALENCE (C(2000),T      ), (C(2001),TF     ), (C(2003),PCNT  )
      EQUIVALENCE (C(2010),STEP   ), (C(2011),KSTEP  ), (C(2020),LCONV )
      EQUIVALENCE (C(2021),KCONV  ), (C(2561),N      ), (C(2662),HMIN  )
      EQUIVALENCE (C(2663),HMAX   ), (C(2664),DER(1)), (C(2765),EL(1))
      EQUIVALENCE (C(2865),EU     ), (C(2965),VAR(1))
      EQUIVALENCE (C(1973),KASE   ), (C(1974),NJ     ), (C(1975),NPT   )
      DIMENSION   DER(101)         , VAR(101)         , EL(100)
      EXTERNAL AUXSUB
      CALL G4
      IF (ABS( T-TF) .LE. 0.01 ) GO TO 20
      IF ( (TF-T) .LT. 0.) GO TO 10
      IF (LCONV .EQ. 2) GO TO 20
      IF (LCONV .EQ. 1) GO TO 10
      IF(DER(1) .LT. 0.) DER(1)=-DER(1)*.5
      RETURN
   10 IF(DER(1) .GT. 0.) DER(1)=-DER(1)*.5
      KCONV = KCONV + 1
      IF(KCONV .GE. 10) GO TO 20
      RETURN
   20 PCNT = 1.0
      IF(STEP .EQ.11.)GOTO 40
      PREDER = DER(1)
      DER(1) = 0.
      NJ=N-1
      NPT=0
      CALL AMRK(AUXSUB)
      DER(1) = PREDER
   40 CALL OUPT3
      KSTEP = 2
      RETURN
      END
```

```
      SUBROUTINE   SUBL2
      COMMON C(3830)
      EQUIVALENCE (C(2461),NOSUB ), (C(2462),SUBNO(1) )
      DIMENSION   SUBNO(99)
      DO 1 I = 1, NOSUB
      J = SUBNO(I)
      GO TO ( 1, 2, 3, 4, 5, 6, 7, 8, 9 ), J
    2 CALL INPT2
      GO TO 1
    3 CALL OUPT2
      GO TO 1
    4 CALL STGE2
      GO TO 1
    5 CALL CNTR2
      GO TO 1
    6 CALL RNDM2
      GO TO 1
    7 CALL AUXA2
      GO TO 1
    8 CALL AUXB2
      GO TO 1
    9 CALL AUXC2
    1 CONTINUE
      RETURN
       END
```

```
      SUBROUTINE   SUBL3
      COMMON C(3830)
      EQUIVALENCE (C(2461),NOSUB ), (C(2462),SUBNO(1) )
      DIMENSION   SUBNO(99)
      DO 1 I = 1, NOSUB
      J = SUBNO(I)
      GO TO ( 1, 2, 3, 4, 5, 6, 7, 8, 9 ), J
    2 CALL INPT3
      GO TO 1
    3 CALL OUPT3
      GO TO 1
    4 CALL STGE3
      GO TO 1
    5 CALL CNTR3
      GO TO 1
    6 CALL RNDM3
      GO TO 1
    7 CALL AUXA3
      GO TO 1
    8 CALL AUXB3
      GO TO 1
    9 CALL AUXC3
    1 CONTINUE
      RETURN
       END
```

```
      SUBROUTINE S1
C**SEEKER MODULE
C
      COMMON C(3830)
      EQUIVALENCE (C(2000),T      )
  101 FORMAT (30H0      TARGET ACQUISITION    T = ,F8.4,
     *        10H   EPS Z = ,1PE11.3,10H   EPS Y = ,1PE11.3)
  102 FORMAT (30H0      PITCH PLANE TRACK     T = ,F8.4,
     *        10H   EPS Z = ,1PE11.3,10H   EPS Y = ,1PE11.3)
  103 FORMAT (30H0      YAW   PLANE TRACK     T = ,F8.4,
     *        10H   EPS Z = ,1PE11.3,10H   EPS Y = ,1PE11.3)
C
C**INPUT DATA
      EQUIVALENCE (C( 445),RLOCK )
      EQUIVALENCE (C( 446),DT    )
      EQUIVALENCE (C( 447),BDB   )
      EQUIVALENCE (C( 448),CFOVZ )
      EQUIVALENCE (C( 449),CFOVY )
      EQUIVALENCE (C( 450),GSX   )
      EQUIVALENCE (C( 451),SEPS  )
      EQUIVALENCE (C( 452),SWP   )
      EQUIVALENCE (C( 453),RBK   )
      EQUIVALENCE (C( 454),GEO   )
      EQUIVALENCE (C( 455),OPTNSK)
      EQUIVALENCE (C(0456),GS    )
      EQUIVALENCE (C(0457),WSL   )
      EQUIVALENCE (C(0458),WSN   )
      EQUIVALENCE (C(0459),WL2   )
C
      EQUIVALENCE (C( 460),ST    )
      EQUIVALENCE (C( 461),CAGE  )
      EQUIVALENCE (C( 462),TKRZ  )
      EQUIVALENCE (C( 463),TKRY  )
      EQUIVALENCE (C( 464),TRKZY )
C
C**INPUTS FROM OTHER MODULES
      EQUIVALENCE (C(0371),RANGE )
      EQUIVALENCE (C(0372),RXBA  )
      EQUIVALENCE (C(0373),RYBA  )
      EQUIVALENCE (C(0374),RZBA  )
      EQUIVALENCE (C(1739),WP    )
      EQUIVALENCE (C(1743),WQ    )
      EQUIVALENCE (C(1747),WR    )
C
C**STATE VARIABLE OUTPUTS
      EQUIVALENCE (C(0408),WLQD  )
      EQUIVALENCE (C(0411),WLQ   )
      EQUIVALENCE (C(0412),WLRD  )
      EQUIVALENCE (C(0415),WLR   )
      EQUIVALENCE (C(0416),WLQSD )
      EQUIVALENCE (C(0419),WLQS  )
      EQUIVALENCE (C(0420),WLRSD )
      EQUIVALENCE (C(0423),WLRS  )
      EQUIVALENCE (C(0424),BTHTGD)
      EQUIVALENCE (C(0427),BTHTG )
      EQUIVALENCE (C(0428),BPSIGD)
      EQUIVALENCE (C(0431),BPSIG )
C
C**OTHER OUTPUTS
```

```
      EQUIVALENCE (C(11),BY)
      EQUIVALENCE (C(12),BZ)
      EQUIVALENCE (C(0403),WLAMQ )
      EQUIVALENCE (C(0407),WLAMR )
      EQUIVALENCE (C(0435),BEPSZ )
      EQUIVALENCE (C(0436),BEPSY )
      EQUIVALENCE (C(0437),WZ    )
      EQUIVALENCE (C(0438),WY    )
      EQUIVALENCE (C(0439),BGDEFL)
      EQUIVALENCE (C( 465),  SDY)
      EQUIVALENCE (C( 466),  SDZ)
      IF(C(899).GT.0.0)RETURN
C
C**DIRECTION COSINES FOR BODY TO PLATFORM TRANSFORMATION
      BTACT = BTHTG
      BPACT = BPSIG
      UCT=COSD(BTACT)
      UST=SIND(BTACT)
      UCP=COSD(BPACT)
      USP=SIND(BPACT)
      UB11 = UCT*UCP
      UB12 = UCT*USP
      UB13 =-UST
      UB21 =-USP
      UB22 = UCP
      UB23 = 0.
      UB31 = UST*UCP
      UB32 = UST*USP
      UB33 = UCT
C
C** CALCULATE TOTAL DEFLECTION OF GIMBALS
      BGDEFL=SQRT(BTHTG**2+BPSIG**2)
C
C**TRANSFORM LOS FROM BODY TO GIMBAL AXES
      RXG = UB11*RXBA+UB12*RYBA+UB13*RZBA
      RYG = UB21*RXBA+UB22*RYBA+UB23*RZBA
      RZG = UB31*RXBA+UB32*RYBA+UB33*RZBA
C
C**LOS ERRORS IN PLATFORM COORDINATES
      BEPSZ = ATAND(-RZG,RXG)
      BEPSY = ATAND( RYG,RXG)
C
C**SEEKER OUTPUT SIGNALS
      IF(C(1976).LE.0.) GO TO 82
      IF(T.LT.(ST-.000001)) GO TO 82
      IF(C(13).LE.0.) GO TO 820
      C(13) = -1.
      ST = T
      C(2664) = DT / AINT(DT / C(2764))
  820 CONTINUE
      ST = ST + DT
C**VIDICON TRACKER
      IF(OPTNSK .LE. 0. ) GO TO 85
      WLAMQ = GEO * BEPSZ
      WLAMR = GEO * BEPSY
      WQP = WLAMQ
      WRP = WLAMR
      GO TO 30
C**QUADRANT TRACKER
   85 CONTINUE
```

```
      IF (RANGE .GT. RLOCK) GO TO 81
      CZ = 2.*BEPSZ/CFOVZ
      CY = 2.*BEPSY/CFOVY
      IF (CZ**2 .GT. 1.-CY**2) GO TO 81
      BZ = SIGN(1.,BEPSZ)
      BY = SIGN(1.,BEPSY)
      TKDB = BDB/2.*(RANGE/32810.)**2
      IF (ABS(BEPSZ).LT.TKDB) BZ = 0.
      IF (ABS(BEPSY).LT.TKDB) BY = 0.
      CALL QD
      IF (CAGE .GT. 0.) GO TO 82
      UZ = BZ
      UY = BY
      CAGE = 1.
      WRITE(6,101) T, BEPSZ, BEPSY
      GO TO 82
   81 BZ = 0.
      BY = 0.
C**SEEKER COMPENSATION
   82 IF(OPTNSK .GT. 0.) GO TO 30
      WLAMQ = BZ * GS
      WLAMR = BY*GS
      WQP = WLAMQ
      WRP = WLAMR
      IF (WSL .LE. 0.) GO TO 83
      WLQD = WLAMQ
      WLRD = WLAMR
      WLQD = WLQD + SEPS
      WLRD = WLRD + SEPS
      WQP = WLQD/WSL + WLQ
      WRP = WLRD/WSL + WLR
      WLAMQ = WQP
      WLAMR = WRP
      IF (WSN .LE. 0.) GO TO 83
      WLQSD = WSN*(WQP - WLQS)
      WLRSD = WSN*(WRP - WLRS)
      WQP = WLQSD/WL2 + WLQS
      WRP = WLRSD/WL2 + WLRS
C**SEEKER SWITCHING LOGIC
   83 IF (CAGE .LE. 0.) GO TO 30
C  PITCH PLANE
   10 IF (TKRZ .GT. 0.) GO TO 20
      IF (BZ*UZ .GE. 0.) GO TO 12
      TKRZ = 1.
      WRITE(6,102) T, BEPSZ, BEPSY
      GO TO 20
   12 WLAMQ = BZ*GSX
      WQP = WLAMQ
      WLQD = 0.
      WLQSD = 0.
      UZ = BZ
C  YAW PLANE
   20 IF (TKRY .GT. 0.) GO TO 30
      IF (BY*UY .GE. 0.) GO TO 22
      TKRY = 1.
      WRITE(6,103) T, BEPSZ, BEPSY
      GO TO 30
   22 WLAMR = BY*GSX
      WRP = WLAMR
      WLRD = 0.
```

```
      WLRSD = 0.
      UY = BY
   30 CONTINUE
C
C**MISSILE BODY RATES IN GIMBAL AXES
      WZ = UB31*WP+UB32*WQ+UB33*WR
      WY = UB21*WP+UB22*WQ+UB23*WR
C
C**GIMBAL COUPLING
      UZK = SWP*(-BTHTG + .1*BPSIG)
      UYK = SWP*(-BPSIG - .1*BTHTG)
      UZK = UZK + SDZ
      UYK = UYK + SDY
C
C**GIMBAL ANGLE DERIVATIVES
      BTHTGD = WQP + UZK - WY
      BPSIGD = WRP + UYK - WZ/UB33
C
      IF (CAGE .GT. 0.) RETURN
      WLAMQ = 0.
      WLAMR = 0.
      WLQD = 0.
      WLRD = 0.
      WLQSD = 0.
      WLRSD = 0.
      BTHTGD = 0.
      BPSIGD = 0.
      RETURN
      END
```

```
      SUBROUTINE S1I
C**SEEKER INIT.MODULE
      COMMON C(3830)
      DIMENSION IZ(50), IY(50), ISNDX(40)
      EQUIVALENCE (C(3634), ISNDX(1)), (C(3512), I3512)
      EQUIVALENCE (C( 470), BTGERR)
      EQUIVALENCE (C( 471), BPGERR)
      EQUIVALENCE (C( 465),  SDY)
      EQUIVALENCE (C( 466),  SDZ)
    1 FORMAT(5X,2HBZ,6X,4(I13,I11)/(13X,4(I13,I11)))
    2 FORMAT(5X,2HBY,6X,4(I13,I11)/(13X,4(I13,I11)))
      EQUIVALENCE (C(11),BY)
      EQUIVALENCE (C(12),BZ)
      EQUIVALENCE(C(2011),KSTEP)
      EQUIVALENCE(C(600),IZ(1))
      EQUIVALENCE(C(650),IY(1))
      DIMENSION IPL(100)
      EQUIVALENCE (C(452),SWP)
      EQUIVALENCE(C(0411),WLQ)
      EQUIVALENCE(C(0415),WLR)
      EQUIVALENCE (C(0419),WLQS)
      EQUIVALENCE (C(0423),WLRS)
      EQUIVALENCE (C(0427),BTHTG)
      EQUIVALENCE (C(0431),BPSIG)
      EQUIVALENCE (C(2561),N)
      EQUIVALENCE (C(2562),IPL(1))
      EQUIVALENCE (C(3504),OPTN4)
      EQUIVALENCE (C(2662)  ,DERSV)
      IPL(N)=424
      IPL(N+1)=428
      IPL(N+2)=408
      IPL(N+3)=412
      IPL(N+4)=416
      IPL(N+5)=420
      N=N+6
      C(411)=0.
      C(415)=0.
      C(419)=0.
      C(423)=0.
       BY=0.
      BZ=0.
      SDY = 0.
      SDZ = 0.
      DO 10 I = 1, I3512
      IDO = I
C
C MONTE CARLO SEEKER OUTPUT STARTING VALUES
C
      IF(ISNDX(I).EQ.11) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.12) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.460) CALL MCARLO (DUM, 1, IDO)
      IF(ABS(BY).GT.0. ) BY = SIGN(1.,BY)
      IF(ABS(BZ).GT.0. ) BZ = SIGN(1.,BZ)
C
C  MONTE CARLO SEEKER POINTING ERROR
C
      IF(ISNDX(I).EQ.470) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.471) CALL MCARLO (DUM, 1, IDO)
C
```

```
C ** MONTECARLO SEEKER DRIFT
      IF(ISNDX(I).EQ.465) CALL MCARLO (DUM, 1, IDO)
      IF(ISNDX(I).EQ.466) CALL MCARLO (DUM, 1, IDO)
C
   10 CONTINUE
      BTHTG = BTHTG + BTGERR
      BPSIG = BPSIG + BPGERR
C
      WLQS=SWP*(BTHTG-BPSIG)
      WLQ=SWP*(BTHTG-BPSIG)
      WLR=SWP*(BTHTG+BPSIG)
      WLRS=SWP*(BTHTG+BPSIG)
      C(13) = -1.
      DERSV=.002
      C(461)=0.
      C(462)=0.
      C(463)=0.
      C(464)=0.
      IF(OPTN4.GT.1.) GO TO 30
      C(461)=1.
      C(462)=1.
      C(463)=1.
      C(464)=1.
   30 CONTINUE
      NI=1
      MI=1
      SET=0.
      DO 200 I=1,50
      IZ(I)=0
  200 IY(I)=0
      RETURN
      ENTRY QD
      IF(SET.GT.0.) RETURN
      IF(NI.GT.50) RETURN
      IF(MI.LE.10) GO TO 100
      NI=NI+1
      MI=1
  100 IZ(NI)=IZ(NI)+INT(BZ+2.)*10**(10-MI)
      IY(NI)=IY(NI)+INT(BY+2.)*10**(10-MI)
      MI=MI+1
      RETURN
      ENTRY S8
      IF(SET.GT.0..OR.KSTEP.NE.2) RETURN
      SET=1.
      WRITE(6,1) (IZ(I),I=1,NI)
      WRITE(6,2) (IY(I),I=1,NI)
      RETURN
      END
```

```
SUBROUTINE TABLE (X,XI,YI,NX,XK,XLABEL,Y)
DIMENSION  XI(NX),YI(NX)
XK = 0.
Y  =   FINTP1 (X,XI,YI,NX,XK,XLABEL)
RETURN
END
```

```
SUBROUTINE TABL2(X,Y,XI,YI,ZI,NX,NY,NXY,XINTER,XLABEL,Z)
DIMENSION XI(NX) ,YI(NY) ,ZI(NXY)
Z=FINTP2(X,Y,XI,YI,ZI,NX,NY,NXY,XINTER,XLABEL)
RETURN
END
```

```
      SUBROUTINE TIMEV(XPDQ)
      DIMENSION IC(6),X(100),Y(100),Y1(2,4),T1(300)
C  DUMMY SUBROUTINE
      ENTRY A2I
      ENTRY A4
      ENTRY A4I
      ENTRY A5
      ENTRY A5I
      ENTRY C2
      ENTRY C2I
      ENTRY C6
      ENTRY C6I
      ENTRY C7
      ENTRY C7I
      ENTRY C8
      ENTRY C8I
      ENTRY C9
      ENTRY C9I
      ENTRY C10
      ENTRY C10I
      ENTRY D3
      ENTRY D3I
      ENTRY D4
      ENTRY D4I
      ENTRY D5
      ENTRY D5I
      ENTRY G1
      ENTRY G1I
      ENTRY G3I
      ENTRY G4I
      ENTRY G5I
      ENTRY G6
      ENTRY G6I
      ENTRY S5
      ENTRY S5I
      ENTRY S6
      ENTRY S6I
      ENTRY S7
      ENTRY S7I
      ENTRY S8I
      ENTRY S9
      ENTRY S9I
      ENTRY S10
      ENTRY S10I
      ENTRY AUXA1
      ENTRY AUXA2
      ENTRY AUXA3
      ENTRY AUXB1
      ENTRY AUXB2
      ENTRY AUXB3
      ENTRY AUXC1
      ENTRY AUXC2
      ENTRY AUXC3
      ENTRY CNTR1
      ENTRY CNTR2
      ENTRY CNTR3
      ENTRY INPT1
      ENTRY INPT2
      ENTRY INPT3
```

```
      ENTRY NORMAL(RX,XL,XU,XM,SG,RN)
      ENTRY OUPT1
      ENTRY PROCES
      ENTRY RANNUM(R,S,T)
      ENTRY RNDM1
      ENTRY RNDM2
      ENTRY RNDM3
      ENTRY STGE1
      ENTRY KIKSET
      ENTRY COUNTV
      ENTRY WRITE
      GOTO 2
      ENTRY MCARLO(R,M,I)
      GOTO 2
      ENTRY AERROR(XL)
      GO TO 2
      ENTRY TERROR(XL)
      CALL EXIT
      ENTRY CEPAS(N,I,IP,XL,S,IC,P)
      ENTRY CEPP(X,Y,N,S,X1,I,IC,IP,P)
      ENTRY NORM(R,X1,XU,XM,S,R1)
      ENTRY KSTEST(Y,N,S,X1,SX,NI)
      ENTRY ZTABLE(Z,F,N)
      ENTRY PPLOT(X,Y,N,C,I,R,T,XB,YB,XL,P)
      ENTRY XLOC(X1,H,I,IN)
      GOTO 2
      ENTRY G2
      ENTRY G2I
      ENTRY S4
      ENTRY S4I
      ENTRY C5
      ENTRY C5I
      ENTRY RESET
      GOTO 2
      ENTRY PLOT4(G,N,Y1,T1,NP,NL,NO)
      ENTRY S2
      ENTRY S3
      ENTRY PLOT2
      ENTRY PLOTN
      ENTRY MCARLX
      ENTRY SUBL1
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WKPLOT
C
C  PRINTER PLOTS OF SPECIFIED VARIABLES AGAINST TIME AND AGAINST
C  EACH OTHER.
C  ..MAXIMUM OF 4 DEPENDENT VARIABLES PLOTTED AGAINST TIME
C  ..IN PAIRED PLOTS, FIRST VARIABLE IN PAIR IS ASSUMED TO BE
C    THE INDEPENDENT VARIABLE.
C
C  POINTS PLACED IN WKU.YORK ON DPC002.  ACTUAL PLOTTING DONE BY
C      NEXT JOB STEP
C
C  VARIABLES:
C
C     NOPLOT-C(2009)-#PLOTS
C     NPLOT-C(1984)-#PLOTS-COMPUTED-???
C     VLABLE(2,15)-C(2325)-8 CHARACTER AXIS NAMES
C     OUTPLT(15)-C(1985)-C ARRAY INDICIES OF VARIABLES TO BE PLOTED
C     GRAPH(300,4)-POINTS TO BE PLOTTED
C     TIME(300)-C(2025)-PLOT TIME INTERVALS
C     OPOINT-C(2023)-#POINTS TO BE PLOTTED
C     PLOTN2-C(1983)-#PAIRED PLOTS-INPUT
C     PPP-C(2005)-WIDTH OF TIME SAMPLE INTERVAL-INPUT
C     PPNT-C(2004)-TIME PLOTTING IS BEGUN-INPUT
C
      COMMON C(3830),GRAPH(300,4)
C
      EQUIVALENCE (C(2009),NOPLOT),(C(1984),NPLOT),(C(1983),PLOTN2),
     .(C(2325),VLABLE(1,1)),(C(1985),OUTPLT(1)),(C(2025),TIME(1)),
     .(C(2023),OPOINT),(C(2005),PPP),(C(2004),PPNT)
      INTEGER OPOINT,NOPLOT,NPLOT,OUTPLT(15)
      DIMENSION VLABLE(2,15),TIME(300)
C
      NAMELIST/TEST/NOPLOT,NPLOT,PLOTN2,VLABLE,OUTPLT,OPOINT,PPP,PPNT,
     .GRAPH
C
      DEFINE FILE 8(1400,80,E,NPOINT)
C
C
C  OUTPUT GRAPH LABELS
      NOPLOT=NPLOT
      NPL2= PLOTN2
C
      WRITE(8'1,10) NOPLOT,OPOINT,NPL2
   10 FORMAT(3I3)
      IF(NOPLOT .EQ. 0) RETURN
C
C  PLACE TIME ARRAY IN FILE
      DO 14 J=1,OPOINT
   14 WRITE(8'NPOINT,11) TIME(J)
   11 FORMAT(E15.7)
C
C  TIME VS. ---  PLOTS
      DO 4 I=1,NOPLOT
      IF(I.GT.4) WRITE(6,6)
      IF(I.GT.4) RETURN
    6 FORMAT('0WARNING..I>4..YOU CANNOT HAVE MORE THAN 4 TIME PLOTS',
     ./,' ', 80('*'))
C PLACE DEPENDENT VARIABLE IN FILE
      DO 5 J = 1,OPOINT
```

```
    5 WRITE(8'NPOINT,11) GRAPH(J,I)
      WRITE(8'NPOINT,2) VLABLE(1,I),VLABLE(2,I)
    2 FORMAT(2A4)
    4 CONTINUE
C
      WRITE(6,8)
    8 FORMAT('OPLOT POINTS PLACED IN WKU.YORK')
      RETURN
      END
```

```
SUBROUTINE ZERO
COMMON C(3830)
EQUIVALENCE (C(1984),NPLOT )
EQUIVALENCE (C(2023),OPOINT)
EQUIVALENCE (C(2361),NOMOD )
EQUIVALENCE (C(2461),NOSUB )
EQUIVALENCE (C(3066),NOLIST)
EQUIVALENCE (C(3167),NOOUT )
INTEGER OPOINT
NOSUB = 0
NOMOD = 0
NOOUT = 0
NOLIST = 0
OPOINT=0
NPLOT=0
RETURN
END
```

REFERENCES

1. Bryson, A.E. Jr. and Ho, Y.C., Applied Optimal Control, Blaisdell, Waltham, Mass., 1969, pp. 148-176.

2. Kim, M. and Grider, K.V., "Terminal Guidance for Impact Attitude Angle Constrained Flight Trajectories," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-9, Nov. 1973, pp. 852-859.

3. Adler, F.P., "Missile Guidance by Three-Dimensional Proportional Navigation," Journal of Applied Physics, Vol. 27, May 1956, pp.500-507.

4. Murtaugh, S.A. and Criel, H.E., "Fundamentals of Proportional Navigation," IEEE Spectrum, Vol.3, Dec. 1966, pp. 76-85.

5. Leistikow, L., McCorkle, R.D., and Rishel, R.W., Optimum Control of Air to Surface Missiles, Rept. AFFDL-TR-66-64, Air Force Flight Dynamics Lab., Wright Patterson Air Force Base, Ohio, March 1967.

6. Stallard, D.V. "Discreto Optimal Terminal Control with Applications to Missile Guidance," IEEE Transactions on Automatic Control, Vol. AC-18, Aug. 1973, pp. 373-376.

7. Breza, M.J., "Terminal Guidance of an Air to Surface Missile Using Optimal Control and Filtering," Master's Thesis, Air Force Institute of Techology, Wright Patterson Air Force Base, Ohio, June 1967.

8. Fifer, S., Analogue Computation, Vol. IV, New York: McGraw-Hill Book Company, 1961 pp. 1087-1176.

9. York, R.J. and Pastrick, H.L., "Optimal Terminal Guidance with Constraints at Final Time," Journal of Spacecraft and Rockets, Vol. 14, No. 6, June 1977, pp. 381-383.

10. Willems, G., Optimal Controllers for Homing Missiles, U.S. Army Missile Command, Redstone Arsenal, Alabama, September 1968, Report No. RE-68-15.

11. Pastrick, H.L., York, R.J., and St. Clair, D.C., "On the Realization of an Optimal Control Law for a Terminally Guided Missile," Proceedings of the Tenth Annual Southeastern Symposium on System Theory, March, 1978 pp. IV-A-38 to IV-A-48.

12. Lewis, C.L., Hooker, W.R., Lee, A.W., Jr, and Harrison, J.S., THAD T-7 Missile Monte-Carlo Terminal Homing Simulation Utilizing ALS, Digital/ Linear and TV Seekers, U.S. Army Missile Command, Redstone Arsenal, Alabama, July, 1976, Report No. RG-7T-2.

13. Lewis, C.L., An Engineering and Programming Guide for a Six Degree of Freedom, Terminal Homing Simulation Program, Guidance and Control Directorate, U.S. Army Missile Command, Redstone Arsenal, Alabama, July, 1973.

14. York, R.J. and Pastrick, H.L., Optimal Control Applications for Missile Systems, (Confidential) Guidance and Control Directorate, U.S. Army Missile Command, Redstone Arsenal, Alabama, February, 1977, Technical Report TG-77-2.

Appendix A. OPTIMAL TERMINAL GUIDANCE WITH CONSTRAINTS AT FINAL TIME

Randy J. York
Western Kentucky University
Bowling Green, Kentucky 42101

Harold L. Pastrick
US Army Missile Command
Redstone Arsenal, Alabama 35809

and

Robert Pervine
Ph. D Candidate
University of Kentucky
Lexington, Kentucky 40506

Kent Pavey
Ph. D Candidate
University of Colorado
Boulder, Colorado 80302

ABSTRACT

A suboptimal terminal guidance law for a tactical guided missile was derived using a system of state equations describing the geometric and dynamic conditions of the missile-target closure. The problem was formulated to minimize a linear quadratic performance index with constraints at final time. Initially, the autopilot implementation assumed an ideal instantaneous response. This assumption was then removed, and the resulting configuration was investigated for various lag times. As would be expected, the control laws derived with the simplifying assumptions removed were much more complex. However, it was noted that in the limiting case, when the autopilot time lag tended to zero, a simpler control law surfaced. Comparisons were made then of the miss distance and attitude angle at impact with varying lag conditions. It was determined that the suboptimal control law was extremely sensitive to various approximations, especially linear subarcs or constants, for the time varying gains.

I. INTRODUCTION

Recent intelligence suggests that the impenetrable nature of heavy armor may be susceptible to missile attacks at a relatively high angle of impact, with respect to the horizon. In many modes of direct encounter, the target may not be reachable with a body pitch attitude angle of the proper magnitude. There are several possible reasons for this condition including lack of energy (fuel), lack of time to maneuver into the more desirable attitude, or lack of control information by appropriate sensors to command the response. This condition has been recognized for some time at the Missile Research and Development Command and consequently there have been attempts to modify trajectory shapes by a variety of predetermined control laws. However, there has been a certain lack of robustness in the solutions obtained over the entire range of conditions anticipated.

This situation motivated a search for optimal solutions to the guidance problem and a study of tradeoffs among the suboptimal candidates which were deemed feasible.

Terminal guidance schemes for tactical missiles may be based on a classical approach, such as a proportional navigation and guidance law [3,4], or on a modern control theoretic approach [5-8]. In the latter, a control law is derived in terms of time-varying feedback gains when formulated as a linear quadratic control problem. A suboptimal terminal guidance system for reentry vehicles, derived using the modern approach, was the basis for the initial work on this problem.

Kim and Grider [2] studied a suboptimal terminal guidance system for a reentry vehicle by placing a constraint on the body attitude angle at impact. Their problem was oriented to a long range high altitude mission. Their scenario was formulated as a linear quadratic control problem with certain key assumptions. The angle of attack of the reentry vehicle was assumed to be small and thus was neglected. Furthermore, the autopilot response was assumed to be instantaneous, i.e., with no lag time attributed to the transfer of input commands to output reaction.

These conditions have been studied in an extension of their earlier work [9]. A formulation is given for a system that has finite time delay. In fact, the increase and decrease in time delay has interesting ramifications on the solution. The angle of attack assumption is investigated, and though not solved analytically in closed form, the system is derived.

There is more than just a passing academic interest in this problem. As suggested previously, the antiarmor role of several Army weapon systems very well may be enhanced by this technique. The reduction to a practical implementation of mechanization will be studied and described in a future paper. This paper, however, summarizes the feasibility of the concept.

II. STATE REPRESENTATION AND PROBLEM FORMULATION

The geometry of the tactical missile-target position is given in Figure A-1. Assuming that the angle of attack is small and can thus be neglected (this assumption will be considered later) and choosing the following set of state variables

$$x = \begin{bmatrix} Y_d \\ \dot{Y}_d \\ A_L \\ \theta \end{bmatrix} \equiv \begin{bmatrix} Y_t - Y_m \\ \dot{Y}_t - \dot{Y}_m \\ A_L \\ \theta \end{bmatrix} . \tag{A-1}$$

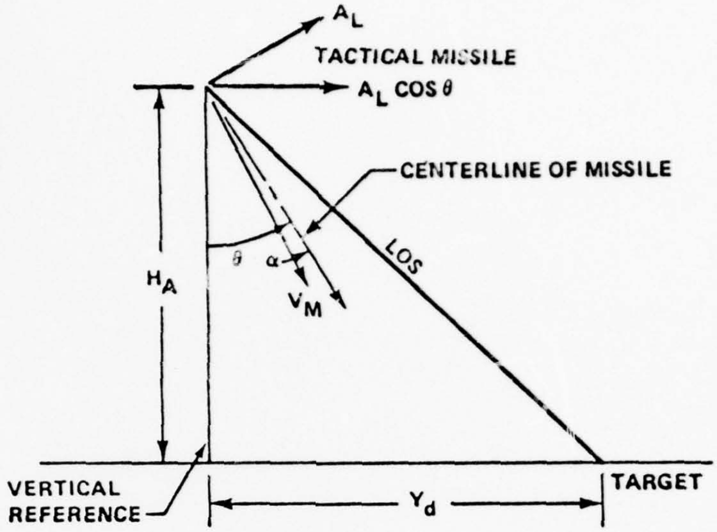

Figure A-1. Geometry of tactical missile target positions.

where

$Y_d \equiv$ the position variable from the missile to the target, projected on the ground

$Y_t \equiv$ the position variable of the target

$Y_m \equiv$ the position variable of the missile projected on the ground

$\dot{Y}_d \equiv$ the derivative of Y_d, the missile to the target velocity projected on the ground

$A_L \equiv$ the lateral acceleration of the missile

$\theta \equiv$ the body attitude angle of the missile

$\alpha \equiv$ the angle of attack of the missile,

the system dynamics can be expressed as

$$\dot{Y}_d = \dot{Y}_d$$

$$\ddot{Y}_d = -A_L \cos \theta$$

$$\dot{A}_L = -\omega_1 A_L + K_1 u \qquad \text{(A-2)}$$

$$\dot{\theta} = K_a u$$

Note that the lag in the autopilot has been represented by a first order lag network

$$\frac{A_L(s)}{u(s)} = \frac{K_1}{s + \omega_1} \qquad \text{(A-3)}$$

where u represents the control. (Previous work [2] assumed immediate response of the autopilot.)

Linearizing about an operating point (i.e., $\cos \theta = b$) and viewing the system in the standard canonical form,

$$\dot{x} = Ax + Bu$$

the result is

$$\begin{bmatrix} \dot{Y}_d \\ \ddot{Y}_d \\ \dot{A}_L \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -b \\ 0 & 0 & 0 & -\omega_1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_d \\ \dot{Y}_d \\ \theta \\ A_L \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ K_1 \\ K_a \end{bmatrix} u \quad . \qquad \text{(A-4)}$$

This optimal control problem will have a controller of the form

$$u = C_Y * Y_d + C_{\dot{Y}} * \dot{Y}_d + C_\theta * \theta + C_{A_L} * A_L \qquad \text{(A-5)}$$

where C_Y, $C_{\dot{Y}}$, C_θ, C_{A_L} are time varying coefficients chosen to minimize the cost functional

$$J = Y_d^2(t_f) + \gamma\theta^2(t_f) + \beta \int_{t_o}^{t_f} u^2(t)\, dt \quad . \qquad \text{(A-6)}$$

Here

$t_f \equiv$ final (impact) time

$t_o \equiv$ initial time

$\gamma, \beta \equiv$ constant weighting factors.

The integral term in the performance index of Equation (A-6) is used to constrain the total expenditure of u. The actual constraints on miss distance and attitude angle at impact are:

$$|Y_d(t_f)| \leq 5 \text{ ft}$$

$$|\theta(t_f)| \leq 5 \text{ deg} \quad .$$

III. PROBLEM SOLUTION

Using the Euler-Langrange formulation, a closed form solution of the controller, u, was obtained as:

$$C_Y = \left\{ -\frac{g\gamma K_a^2 (t - t_f)^2}{2} + \left[(g\beta) + \left(\frac{g\gamma K_a^2}{\omega_1} \right) e^{\omega_1 (t-t_f)} \right] (t - t_f) + \left(\beta + \frac{\gamma K_a^2}{\omega_1} \right) \frac{g}{\omega_1} \left(1 - e^{\omega_1 (t-t_f)} \right) \right\} \Big/ \Delta$$

$$C_{\dot{Y}} = \left\{ -\frac{g}{\omega_1} \left(1 - e^{\omega_1 (t-t_f)} \right) \left(\beta + \frac{\gamma K_a^2}{\omega_1} \right) (t - t_f) + \left[-g\beta - \left(\frac{g\gamma K_a^2}{\omega_1} \right) e^{\omega_1 (t-t_f)} \right] (t - t_f)^2 + \frac{g\gamma K_a^2}{2} (t - t_f)^3 \right\} \Big/ \Delta$$

$$C_\theta = \left\{ -\frac{\gamma g^2 K_a (t - t_f)^3}{6} - \frac{\gamma g^2 K_a}{\omega_1}\left(1 - e^{\omega_1 (t-t_f)}\right)\frac{(t - t_f)^2}{2} \right.$$

$$- \frac{g^2 \gamma K_a (t - t_f)}{\omega_1^2} - \frac{\gamma K_a g^2}{\omega_1^3}\left(1 - e^{\omega_1(t-t_f)}\right)^2 - \gamma K_a \beta$$

$$\left. + \frac{\gamma K_a g^2}{2\omega_1^3}\left(e^{2\omega_1(t-t_f)} - 1\right)\right\} \Big/ \Delta$$

$$C_{A_L} = \left\{ \frac{g\beta}{\omega_1^3}\left(\beta + \frac{\gamma K_a^2}{\omega_1}\right)\left(1 - e^{\omega_1(t-t_f)}\right)^2 + \left(1 - e^{\omega_1(t-t_f)}\right) \right.$$

$$* \left[2\beta\omega_1 + \gamma K_a^2 * \left(1 + e^{\omega_1(t-t_f)}\right)\right] * (t - t_f) \tag{A-7}$$

$$+ \left[\beta\omega_1^2 + \frac{\gamma K_a^2 \omega_1}{2} \cdot \left(3e^{\omega_1(t-t_f)} - 1\right)\right] * (t - t_f)^2$$

$$\left. + \left(-\gamma K_a^2 \omega_1^2\right) * \frac{(t - t_f)^3}{2} \right\} \Big/ \Delta$$

where

$$g \equiv \frac{-b \cdot K_1}{\omega_1}, \quad t \equiv \text{time}$$

and

$$\Delta \equiv \beta^2 - \left(\frac{g^2\beta}{2\omega_1^3}\right) \cdot \left(e^{2\omega_1(t-t_f)} - 1\right) - \frac{\gamma g^2 K_a^2}{\omega_1^4} \cdot \left(1 - e^{\omega_1(t-t_f)}\right)^2$$

$$+ \frac{(t - t_f)}{2}\left[- \frac{2g^2\beta}{\omega_1^2} \cdot \left(1 - 2e^{\omega_1(t-t_f)}\right) - 2\gamma K_a^2 \beta - \frac{\gamma g^2 K_a^2}{\omega_1^3} \right.$$

$$\left. \cdot \left(5 - 4e^{\omega_1(t-t_f)} - e^{2\omega_1(t-t_f)}\right)\right]$$

$$+ (t - t_f)^2 * \left[- \frac{\gamma g^2 K_a^2}{\omega_1^2} \cdot \left(1 + e^{\omega_1 (t-t_f)}\right) - \frac{g^2 \beta}{\omega_1} \right]$$

$$+ \frac{(t - t_f)^3}{3} [-\beta g^2] + \frac{(t - t_f)^4}{12} \left[g^2 \gamma K_a^2 \right] \quad .$$

The coefficients are shown in Figures A-2, A-3, A-4, and A-5 for following parameter values: $\omega_1 = 5$, $K_1 = 5$, $K_a = 0.0005$, $\gamma = 3823$, $\beta = 0.0000694$, $t_f = 7.68$. The jinks near impact time are characteristic of the choice of state variables.

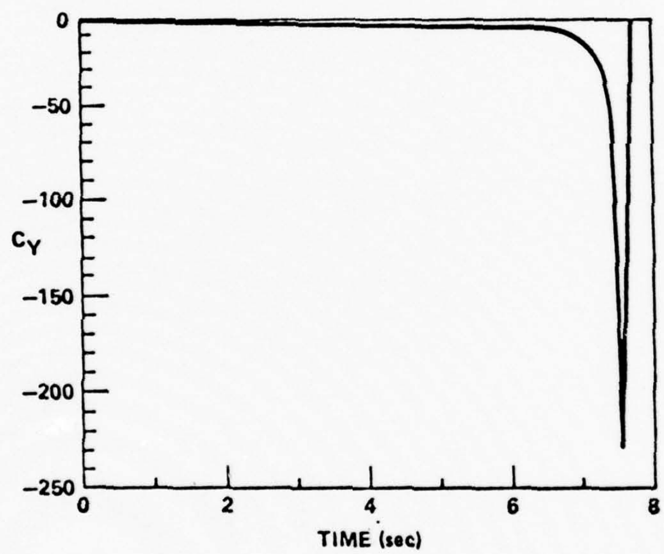

Figure A-2. Time varying optimal history of coefficient C_Y

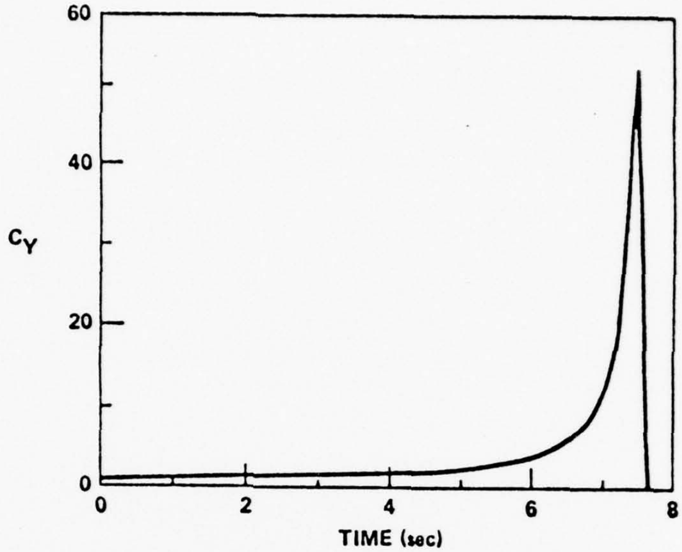

Figure A-3. Time varying optimal history of coefficient $C_{\dot{Y}}$.

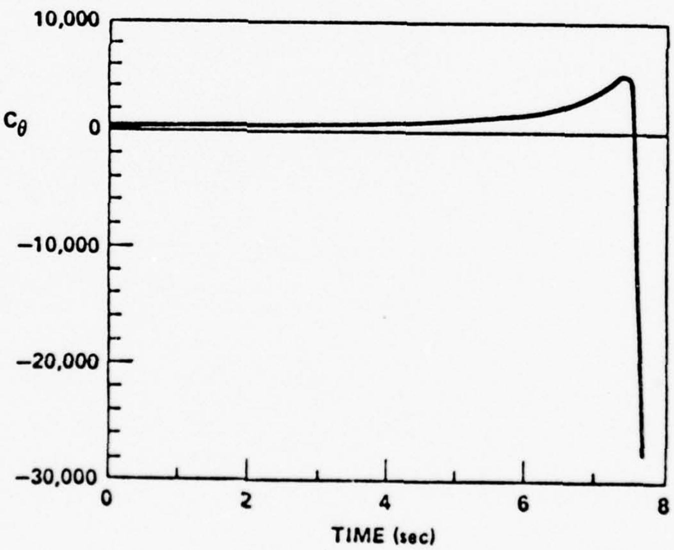

Figure A-4. Time varying optimal history of coefficient C_θ.

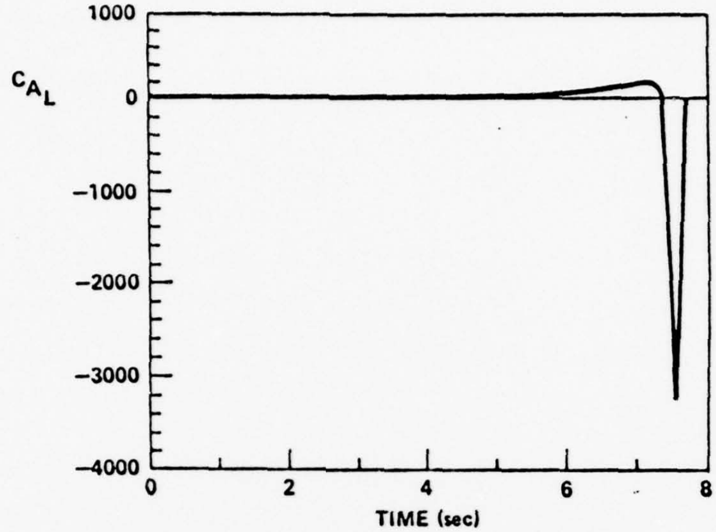

Figure A-5. Time varying optimal history of coefficient $\left\{C_{A_L}\right\}$.

IV. EFFECTIVENESS OF THE CONTROL LAW

In an earlier work [2], a control law of the form

$$u = C_Y * Y_d + C_{\dot{Y}} * \dot{Y}_d + C_\theta * \theta$$

was derived under the assumption of zero autopilot lag (i.e. $A_L = K_1/\omega_1\ u$). The coefficients obtained were

$$C_Y = \frac{\left[-\beta g(t_f - t) - g\gamma K_a^2 (t_f - t)^2/2\right]}{\Delta}$$

$$C_{\dot{Y}} = \frac{\left[-\beta g(t_f - t)^2 - g\gamma K_a^2 (t_f - t)^3/2\right]}{\Delta}$$

$$C_\theta = \frac{\left[-\beta\gamma K_a + \gamma K_a g^2 (t_f - t)^3/6\right]}{\Delta}$$

where

$$g \equiv \frac{-bK_1}{\omega_1}$$

and

$$\Delta \equiv \beta^2 + \gamma\beta K_a^2(t_f - t) + \frac{\beta g^2(t_f - t)^3}{3} + \frac{\gamma g^2 K_a^2(t_f - t)^4}{12} \qquad \text{(A-8)}$$

Note the relationship between the two control laws. As lag in the autopilot tends to zero i.e., $\omega_1 \rightarrow \infty$ in Equation (A-7), the control law of Equation (A-8) surfaces as the limiting case of the new control law.

Even though acceptable performace is obtained using the control law described by the coefficients in Equation (A-8) there is some question regarding the sensitivity of performance due to lag. In particular, this question arises: at what point does performance degenerate to justify the more complex control law to achieve the given performance constraints? This is partially answered by referring to Figure A-6, which is a plot of miss distance versus lag for the two control laws, and Figure A-7 which is attitude angle at impact versus lag.

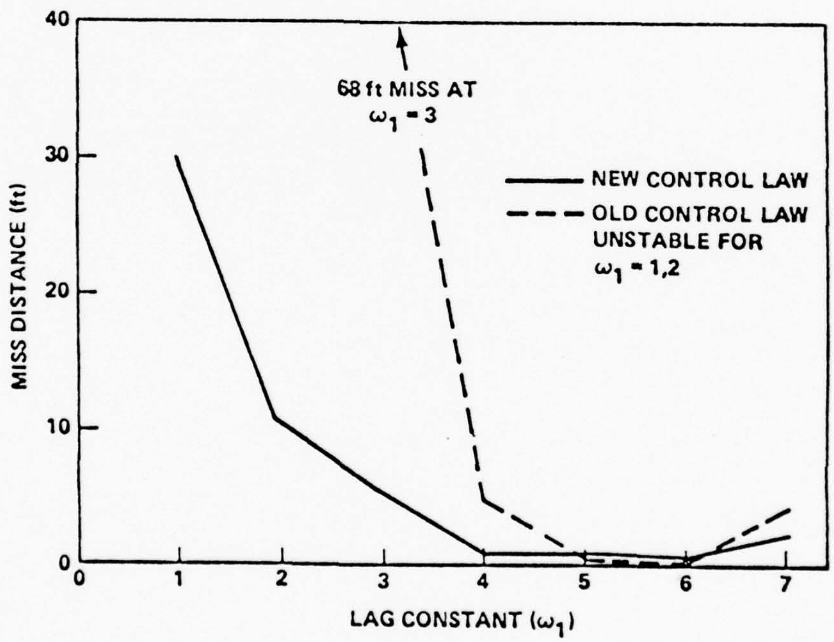

Figure A-6. A comparison of miss distance performance.

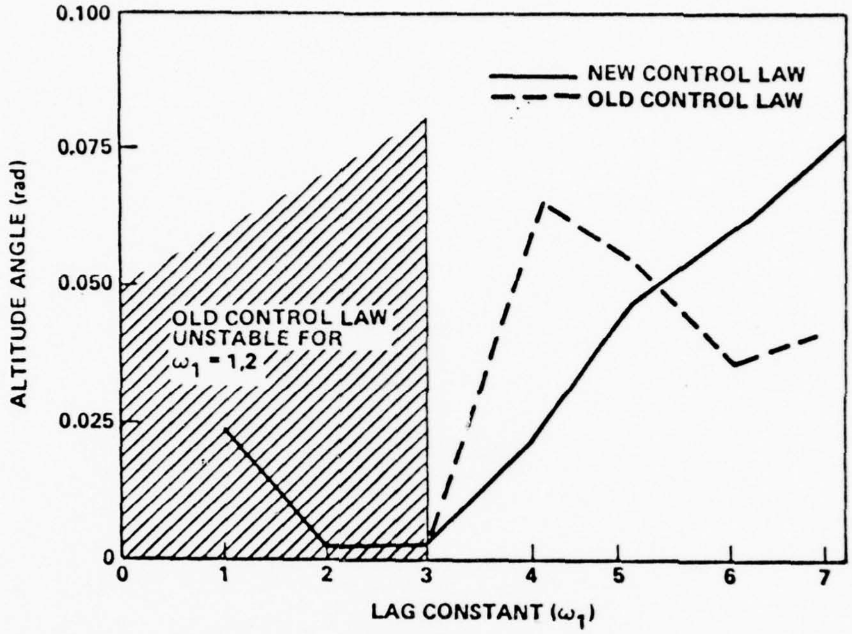

Figure A-7. A comparison of impact angle performance.

For lag constraints of $\omega_1 = 1, 2$, the simulation with the control law of Equation (A-8) becomes unstable. This is not true with the law described by Equation (A-7). For $\omega_1 = 3$, a miss distance of 68 ft was obtained while the new control law had a miss distance of 6.1 feet. Acceptable performance for Equation (A-8) is not obtained until $\omega_1 > 4$. It should be noted that the parameters in that control law were chosen for a nominal lag value of $\omega_1 = 5$. Should the amount of lag in the tactical missile autopilot vary, and not be approximated closely a priori, then the more complicated control law needs to be implemented. In other words, the control law of Equation (A-8) performs adequately only for lag near the assumed nominal value.

The control law implementation was investigated for adaptibility to approximation signals from physically realizable sources. The coefficients C_Y, $C_{\dot{Y}}$, were approximated by linear functions pieced together at three break points. Performance turned out to be too sensitive to the approximation error, and the performance constraint could not be met. The number of linear segments used was increased, but acceptable performance still was not achieved. It was concluded that successful implementation would require more than merely linear approximations to the time varying coefficients.

The angle of attack probably cannot be ignored for the larger tactical missiles. For such a missile, the system of equations should include the angle of attack α. In addition, because it is feasible to achieve only a small angle of attack at impact, a reasonable performance index to be minimized would seem to be:

$$J = C_1 Y_d^2(t_f) + C_2 \theta^2(t_f) + C_3 \alpha^2(t_f) + C_4 \int_{t_o}^{t_f} u^2(t)\, dt \qquad \text{(A-9)}$$

To produce a formulation of the problem which incorporates the angle of attack α, it is assumed that $\dot{\alpha}$ can be expressed as a linear combination of the control u and the attitude rate $\dot{\theta}$. Assuming motion only in the pitch plane, then

$$\dot{\theta} = Q \text{ (pitch rate)}$$

$$\ddot{\theta} = \dot{Q} = (TAB_2 + TCB_2)/I_2$$

where TAB_2 is the pitching moment coefficient due to angle of attack and pitch rate, TCB_2, is the pitching moment coefficient due to fin deflection. Now,

$$TAB_2 = -qSd(C_{m_\alpha} \cdot \alpha + C_{m_\theta} \cdot Q)$$

or

$$TAB_2 = L_1 \alpha + L_2 \dot{\theta} \quad ,$$

where

$$L_1 = -q \cdot S \cdot d \cdot C_{m_\alpha} / I_2$$

$$L_2 = -q \cdot S \cdot d \cdot C_{m_\theta} / I_2$$

$q \equiv$ dynamic pressure

$S \equiv$ missile reference area

$d =$ missile reference dimension.

Also, noting pitch fin deflection (δ_q) is equivalent to control (u)

$$TCB_2 = q \cdot S \cdot d \cdot \cdot C_{m_\delta} \cdot u \quad .$$

Therefore,

$$\ddot{\theta} = L_1\alpha + L_2\dot{\theta} + L_3 u$$

where

$$L_3 = q \cdot S \cdot d \cdot C_{m_\delta} / I_2 \quad .$$

Figure A-1, yields the following system for the state variables Y_d, $\dot{Y}_d$, A_L, θ, $\dot{\theta}$, α:

$$\dot{Y}_d = \dot{Y}_d$$

$$\ddot{Y}_d = -A_L \cos (\theta - \alpha)$$

$$\dot{A}_L = -\omega_1 A_L + K_1 u$$

$$\dot{\theta} = \dot{\theta}$$

$$\ddot{\theta} = L_1\alpha + L_2\dot{\theta} + L_3 u$$

$$\dot{\alpha} = K_3 u + K_4\dot{\theta}$$

This system can be linearized about an operating point, i.e., $b = \cos (\theta - \alpha)$. Even so, the control problem with the cost functional in Equation (A-9) does not lend itself readily to a closed form solution. Future work is anticipated using computer augmented algorithms.

REFERENCES

1. Bryson, A. E. Jr., and Ho, Y. C., Applied Optimal Control, Blasdell Publishing Company, Waltham, Massachusetts, 1969, p. 66.

2. Kim, M. and Grider, K. V., "Terminal Guidance for Impact Attitude Angle Constrained Flight Trajectories," IEEE Transaction on Aerospace and Electronic Systems, Vol. AES-9, No. 6, November 1973.

3. Adler, F. P., "Missile Guidance by Three-Dimensional Proportional Navigation," Journal of Applied Physics, Vol. 27, pp. 500-507, May 1956.

4. Murtaugh, S. A. and Criel, H. E., "Fundamentals of Proportional Navigation," IEEE Spectrum, Vol. 3, pp. 75-85, December 1966.

5. Leistikow, L., McCorkle, R. D., Rishel, R. W., Optimum Control of Air to Surface Missiles, Report No. AFFDL-TR-66-64, Air Force Flight Dynamics Laboratory, Wright Patterson Air Force Base, Ohio, March 1967.

6. Breza, M. J., Terminal Guidance of an Air to Surface Missile Using Optimal Control and Filtering, Master's Thesis, Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio, June 1967.

7. Willems, G., Optimal Controllers for Homing Missiles, Report No. RE-TR-68-15, US Army Missile Command, Redstone Arsenal, Alabama, September 1968.

8. Stallard, D. V., Classical and Modern Guidance of Homing Interceptor Missiles, Report No. P247, Raytheon Co; Missile Systems Div., Bedford, Massachusetts, April 1968.

9. Pastrick, H. L. and York, R. J., "Optimal Attitude Control for a Guided Missile Application," Proceedings 1976 Conference on Information Sciences and Systems, John Hopkins University, Baltimore, Maryland, April 1976.

Appendix B: Four-State Simulation Listing

```
      FUNCTION KGCTRL(GAMMA,BETA,NBASE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL *4 KGCTRL,GAMMA,BETA
      REAL *8 K1,KA
      COMMON/CONSTR/CQUANT
      COMMON/YORK1/DTRK,
C
C****INTEGRALS FOR RUNGK
     #        YD,YV,THETA,HA
     #       ,AL
C****DERIVATIVES FOR RUNGK
     #       ,DYD,DYV,DTHETA,DHA
     #       ,DAL
C
     #   ,NX,IYORK,KUTTA
C
      SIN(X)=DSIN(X)
      COS(X)=DCOS(X)
C
C*****KIM-GRIDER CONTROL LAW
      C1Y(OG)=(BETA*G*OG - G*GAMMA*KA**2*OG**2/2.)/DEL
      C1YD(OG)=(-BETA*G*OG**2+G*GAMMA*KA**2*OG**3/2.)/DEL
      C1THET(OG)=(-BETA*GAMMA*KA-GAMMA*KA*G**2*OG**3/6.)/DEL
      C1AL(OG)=0.
      VMI(T) = -39.4 * T + 1080.
777   CONTINUE
C998  FORMAT(3F10.0)
C -
C****** INPUT VALUES FOR GAMMA AND BETA
C     READ(5,998) GAMMA,BETA
C
      IF(GAMMA.EQ.0.) STOP
C     PRINT 100,BETA,GAMMA
C 100 FORMAT(1H1,1X,'BETA=',D14.6,2X,'GAMMA=',D14.6)
      TF=7.0
      TIME=0.
      DT=1./128.
      DTRK=DT
      NX=5
      K1=1.00
      W1=9.80
      KA=.0005
      JPRI=10
      YD=5000.0
      VM=VMI(TIME)
      VT=0.
      AL=20.
C THDEG = LAUNCH ANGLE          DELO = DESIRED IMPACT ANGLE
      THDEG=87.5
      DELO = 45.0
      DELRAD=DELO/57.296
C   B = COS(DELO)
      B = 0.707107
      THETA=(THDEG-DELO)/57.296
C     PRINT 105,GAMMA,DELO,THDEG,B
C 105 FORMAT(' GAMMA=',F10.4,2X,'DELO=',F10.4,2X,'THETA=',F10.4,2X,'B=',
C    AF10.4)
      YV=VT-VM*SIN(THETA+DELRAD)
      HA=1000.0
```

```
      J=20
      IF (NBASE .EQ. 0 ) WRITE(6,1319) TF, K1, W1, YD,
     #HA,VM,B,THDEG,DELO,KA
 1319 FORMAT('OINITIAL VALUES OF TF',G10.4,/' K1 ',G10.5,' W1 ',G10.5,'
     * YD ',G10.4,' HA ',G10.4,' VM ',G10.4,' B ',G10.4,/' INITIAL THDEG
     #', G10.5, 'IMPACT ANGLE DESIRED', G10.5,/,' KA',G10.5/'0')
    2 CONTINUE
      J=J+1
      THDEG=THETA*57.296
      DELTH=THDEG-DELO
      XM=1.E4-YD
      DO 10 KUTTA=1,4
      VM = VMI(TIME)
      G=-K1*B/W1
      TGO=-HA/(VM*COS(THETA+DELRAD))
      DEL=BETA**2-GAMMA*BETA*KA**2*TGO-BETA*G**2*TGO**3/3.+GAMMA*G**2*KA
     1**2*TGO**4/12.
      CY=C1Y(TGO)
      CYD=C1YD(TGO)
      CT=C1THET(TGO)
      CAL=C1AL(TGO)
      U=CY*YD+CYD*YV+CT*THETA+CAL*AL
      DYD=YV
      DYV=-AL*COS(THETA+DELRAD)
      DTHETA=KA*U
      DHA=-VM*COS(THETA+DELRAD)
      DAL=-W1*AL+K1*U
      GOTO(30,50,60,40),KUTTA
   30 CONTINUE
   60 TIME=TIME+.5*DT
   40 CONTINUE
   50 CALL RUNGK
   10 CONTINUE
      IF(DABS(YD) .LT.50.) DTRK=1./256.
   35 IF (J.LT.JPRI) GO TO 3
   36 CONTINUE
C     PRINT 101,TIME,YD,YV,THETA
  101 FORMAT('OTIME==',F10.4,4X,'REL DIST. YD=',D14.6,'REL. VEL. YV=',
     1D14.6,4X,'THETA=',D14.6)
C     PRINT 102,HA,AL
  102 FORMAT('  VERT. HEIGHT=',D14.6,4X,'AL=',D14.6)
C     PRINT 103,CY,CYD,CT,CAL,U
C 103 FORMAT(1X,'C1Y=',D14.6,4X,'C1YD=',D14.6,4X,'C1THETA=',D14.6,4X,
C    1'C1AL==',D14.6,4X,'U=',D14.6)
      J=0
    3 IF(TIME.GT.TF+.2)GOTO 8
      IF(HA.LT.0.)GOTO7
      GO TO 2
    7 CONTINUE
      THDEG=THETA*57.296
C  SET FUNCTION  AND CONSTRAINT VALUES
      KGCTRL = YD**2 + THDEG**2
C     CQUANT=0.0
      WRITE(6,1313) KGCTRL,YD,  THDEG, GAMMA, BETA, CQUANT
 1313 FORMAT('OFUNCTION VALUE',G15.7,/' MISS DISTANCE',G15.7,'ANGLE:',
     $G15.7, / '   GAMMA', G15.7, '  BETA', G15.7,' CQUANT',G15.7,/,'0',
     .60('*'))
      RETURN
C     PRINT 101,TIME,YD,YV,THDEG
C     PRINT 102,HA,AL
```

```
    8 PRINT 405
  405 FORMAT(' TIME EXCEEDS TF')
      WRITE(6,1313) KGCTRL,YD,  THDEG, GAMMA, BETA, CQUANT
      WRITE(6,1314)
 1314 FORMAT('ORUN TERMINATED',/,4(' ',80('*'),/))
      RETURN
      END
      SUBROUTINE RUNGK
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(5),DX(5),XA(5),DXA(5)
      COMMON/YORK1/DT,X,DX
     #   ,NX,IYORK,KUTTA
      GO TO (10,30,50,70),KUTTA
   10 DO 20 I=1,NX
      XA(I)=X(I)
      DXA(I)=DT*DX(I)
   20 X(I)=X(I)+.5*DXA(I)
      RETURN
   30 TDT=2.*DT
      HDT=.5*DT
      DO 40 I=1,NX
      DXA(I)=DXA(I)+TDT*DX(I)
   40 X(I)=XA(I)+HDT*DX(I)
      RETURN
   50 DO 60 I=1,NX
      VDT=DT*DX(I)
      DXA(I)=DXA(I)+2.*VDT
   60 X(I)=XA(I)+VDT
      RETURN
   70 DO 80 I=1,NX
   80 X(I)=XA(I)+(DXA(I)+DT*DX(I))/6.
      RETURN
      END
```

Appendix C: Determination of Optimal Control Parameters:

An Application of Mathematical Programming

by

Drs. Daniel C. St. Clair and Randy J. York

Western Kentucky University

Introduction

The implementation of a control law in a simulation study requires that the researcher determine apriori the value of several control law parameters. Many of these parameters such as height, weight, distance, velocity, etc. are determined by particular test requirements. However, there may also be a set of parameters which act as constants in the simulation but whose values are difficult to determine apriori. This paper suggests several easy to implement mathematical programming algorithms which can be used to produce reliable estimates of such parameters.

Each algorithm presented attempts to solve the mathematical programming problem

$$\min J(\bar{\alpha}) \tag{C-1}$$

where $\bar{\alpha}$ represents the vector of parameters to be determined and $J(\bar{\alpha})$ is the scalar valued function being minimized. In the application at the end of the paper,

$$J(\bar{\alpha}) = Y_d^{\,2}(\bar{\alpha}) + (\theta(\bar{\alpha})-\theta_f)^2 \tag{C-2}$$

where $Y_d(\bar{\alpha})$ is the missile miss distance, $\theta(\bar{\alpha})$ is the attitude angle at impact, and θ_f is the desired attitude angle at impact. In that example, each determination of $J(\bar{\alpha})$ required execution of the control law simulation program.

In the next section, the computational approach to solving (1) is given along with the description of several nonlinear programming algorithms. References to both original works and works containing additional test results

are provided to assist the reader in easy implementation of the algorithms. The third section describes the application of the Hooke and Jeeves algorithm to a specific control problem.

Review of Mathematical Programming Algorithms

The general approach to solving the nonlinear programming problem

$$\min J(\bar{\alpha}) \qquad (1\ \text{bis})$$

is to proceed from a set of parameter estimates $\bar{\alpha}_{i-1}$ to a better set of parameter estimates $\bar{\alpha}_i$ by using the equation

$$\bar{\alpha}_i = \bar{\alpha}_{i-1} + t_1\, \Delta\bar{\alpha}_i. \qquad \text{(C-3)}$$

The scalar step length t_i as well as the direction vector $\Delta\bar{\alpha}_i$ are determined by the strategy used in devising the particular method under consideration. Some algorithms choose t_i by performing a one-dimensional search, such as those described at the end of this section, along the $\Delta\bar{\alpha}_i$ direction. One area of current research favors omitting the step length search by making appropriate choices for $\Delta\bar{\alpha}_i$, for example see [1,2]. Other algorithms [3,4] select the direction of search and the step length simultaneously. The point $\bar{\alpha}_i$ will be accepted as the desired result if either of the following convergence criteria are met, [5] viz.

$$|J(\bar{\alpha}_i)| < \varepsilon_1 \ , \quad ||\bar{\alpha}_i - \bar{\alpha}_{i-1}|| < \varepsilon_2 \qquad \text{(C-4)}$$

where ε_1 and ε_2 are arbitrarily small and $||\cdot||$ represents the Euclidian norm. If this criteria is not met, the algorithm is repeated.

The development of many nonlinear programming algorithms rely heavily on the quadratic approximation of the objective function $J(\bar{\alpha})$ provided by the first three terms of the Taylor series.

$$J(\bar{\alpha}_{i-1} + \Delta\bar{\alpha}_i) \simeq J(\bar{\alpha}_{i-1}) + \nabla J^T_{i-1}\Delta\bar{\alpha}_i + \frac{1}{2}\, \Delta\bar{\alpha}^T_i H_{i-1}\, \Delta\bar{\alpha}_i \qquad \text{(C-5)}$$

where $\nabla\overline{J}_{i-1}$ is the gradient vector and H_{i-1} is the real symmetric Hessian matrix. Both $\nabla\overline{J}_{i-1}$ and H_{i-1} are evaluated at $\overline{\alpha}_{i-1}$.

Algorithms are classified as 1.) direct search methods, 2.) methods using first partial derivatives, and 3.) methods using second partial derivatives contingent on whether the algorithm in question uses no terms, the first two terms, or all three terms respectively of equation (5). Since the calculation of partial derivatives is not feasible for the type of problems being considered here, the algorithms discussed will be limited to those of the direct search type.

The development of direct search algorithms has been guided by extensive experience and by thinking of the problem as that of following valleys down the side of a mountain. These intuitively developed algorithms obtain the direction vector $\Delta\overline{\alpha}_i$ of equation (3) by observing function values at previous points. A one-dimensional search in the $\Delta\overline{\alpha}_i$ direction is used to determine the step length t_i (see equation (3)).

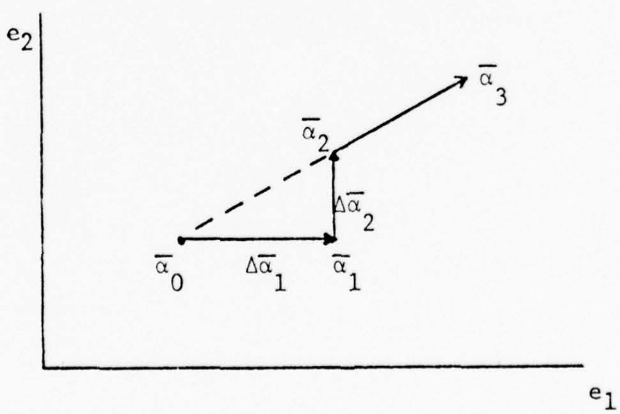

Figure: Hooke - Jeeves Algorithm for n = 2

One simple, but very effective, algorithm is Hooke and Jeeves' Pattern Search [6]. This algorithm consists of a series of "exploratory moves" followed by a "pattern move," see the figure above. Beginning at $\overline{\alpha}_o$, a one-dimensional search is made in each of the n coordinate directions. Thus, for

these exploratory moves, the $\Delta\overline{\alpha}_i$ of equation (3) are the n coordinate directions. The selection of $\overline{\alpha}_n$ completes the first set of exploratory moves. The one-dimensional search done at each step insures that

$$J(\overline{\alpha}_i) \leq J(\overline{\alpha}_{i-1}).$$

The next move is called a pattern move and is made by choosing

$$\overline{\alpha}_{n+1} = \overline{\alpha}_n + (\overline{\alpha}_n - \overline{\alpha}_o). \qquad \text{(C-6)}$$

Since $J(\overline{\alpha}_{n+1})$ is not usually evaluated, it may be the case that

$$J(\overline{\alpha}_{n+1}) > J(\overline{\alpha}_n).$$

A second set of exploratory moves then begins at $\overline{\alpha}_{n+1}$ and proceeds to produce the point $\overline{\alpha}_{2n+1}$. If

$$J(\overline{\alpha}_{2n+1}) \leq J(\overline{\alpha}_n)$$

a pattern move is made from $\overline{\alpha}_{2n+1}$. Otherwise, one returns to $\overline{\alpha}_n$ and begins a new set of exploratory moves.

The pattern moves take large steps along valleys while exploratory moves lead back down to the valley floor. Some programmers modify equation (6) so that $\overline{\alpha}_{n+1}$ is selected as

$$\overline{\alpha}_{n+1} = \overline{\alpha}_n + t_n(\overline{\alpha}_n - \overline{\alpha}_o)$$

where t_n is a steplength chosen so

$$J(\overline{\alpha}_{n+1}) \leq J(\overline{\alpha}_n).$$

Either version is easy to program and requires only function values. Hooke and Jeeves' algorithm is particularly attractive for use in minimizing functions having long curved valleys.

Rosenbrock's algorithm [7] subscribes to the same general "valley following" philosophy as does Hooke and Jeeves' algorithm, but Rosenbrock updates the set of search directions by adding each successful new search direction to the set while removing the "oldest" search direction from the set.

Given a set of n mutually orthogonal directions $\Delta\bar{\alpha}_i$, Rosenbrock's method computes $\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_n$ from an initial estimate $\bar{\alpha}_0$ by using equation (3). A simple linear search is used to determine t_i. This sequence of steps is repeated until nonzero values are found for all t_i, $i = 1, 2, \ldots, n$.

After every n iterations, the set of n search directions $\Delta\bar{\alpha}_i$ is replaced by the set

$$\Delta\bar{\alpha}_k^* = \sum_{i=k}^{n} t_i \, \Delta\bar{\alpha}_i$$

where $k = 1, 2, \ldots, n$. This new set of directions is orthogonalized by the familiar Gram-Schmidt orthonormalization process [8] and the process is repeated. Hence, the method aligns the first search direction along the valley which has just produced a favorable reduction in the value of the objective function. A further development of this idea has been pursued by Davies, Swann, and Campey [9].

An algorithm by M.J.D. Powell [10] is designed to search in such a way that the directions of search $\Delta\bar{\alpha}_i$ are mutually conjugate to the Hessian approximation of inequality (5), viz.

$$\Delta\bar{\alpha}_k^T H_i \, \Delta\bar{\alpha}_j = 0 \qquad \text{for} \qquad k \neq j \text{ and } k, j \leq i.$$

When $J(\bar{\alpha})$ is quadratic, i.e. inequality (5) becomes an equality, it can be shown [11] that the minimum will be found in n steps.

The iterative procedure for Powell's algorithm is given below.

1.) For $i = 1, 2, \ldots, n$, compute

$$\bar{\alpha}_i = \bar{\alpha}_{i-1} + t_i \, \Delta\bar{\alpha}_i$$

where t_i is chosen such that $J(\bar{\alpha}_i) \leq J(\bar{\alpha}_{i-1})$.

2.) Set $\Delta = \max_{1 \leq i \leq n} (J(\bar{\alpha}_{i-1}) - J(\bar{\alpha}_i))$. Let k be the value of i which produces this maximum.

3.) Calculate $J_3 = J(2\bar{\alpha}_n - \bar{\alpha}_0)$ and define $J_1 = J(\bar{\alpha}_0)$ and $J_2 = J(\bar{\alpha}_n)$.

4.) If either $J_3 \geq J_1$ or $(J_1 - 2J_2 + J_3)(J_1 - J_2 - \Delta)^2 \geq \frac{1}{2}\Delta\,(J_1 - J_3)^2$ use the old directions $\Delta\bar{\alpha}_i$, $i = 1, 2, \ldots, n$ and repeat the

algorithm using $\bar{\alpha}_n$ as the starting point.

5.) If neither condition in Step 4 holds, replace $\Delta\bar{\alpha}_k$ with $\Delta\bar{\alpha}_k = (\bar{\alpha}_n - \bar{\alpha}_0)$. Pick the new starting point $\bar{\alpha}_0^*$ so that

$$\bar{\alpha}_0^* = \bar{\alpha}_n + t_k \Delta\bar{\alpha}_k$$

where t has been calculated to insure

$$J(\bar{\alpha}_0^*) \leq J(\bar{\alpha}_n).$$

Steps 1-5 are repeated until the termination criteria are met (see (4)). Each new $\Delta\bar{\alpha}_k$ produced in Step 5 will be conjugate to those previously produced by this step. The initial direction vectors $\Delta\bar{\alpha}_i$, i = 1, 2, ..., n are usually chosen as the coordinate directions.

In certain test cases, Powell observed that his algorithm failed to compute a new $\Delta\bar{\alpha}_k$ when such computation was needed to preserve the linear independence of the current search directions. Such behavior occurs when the valley of the function becomes quite narrow and elongated and $\Delta\bar{\alpha}_k$ repeatedly fails to be updated by Step 5. Zangwill [12] incorporated periodic coordinate searches in Powell's algorithm to resolve this problem.

While there are many other good direct search algorithms available, the ones presented here represent methods which are easily programmed and which require a minimum background in nonlinear optimization. The reader is referred to Himmelblau [5] and to Lessman [11] for a complete discussion of methods, programming details, and examples. We conclude this section with a comment about one-dimensional search algorithms.

To determine the scalar t_i in equation (3) any algorithm may be used which produces a scalar t_i such that

$$J(\bar{\alpha}_{i-1} + t_i \Delta\bar{\alpha}_i) \leq J(\bar{\alpha}_{i-1}).$$

While any t_i satisfying this criteria is acceptable, the greater the reduction in the function value, the better is the performance of the entire algorithm.

Popular one-dimensional search algorithms include the Fibonacci search and the quadratic fit algorithms. These algorithms as well as others are described in Himmelblau [5], Lessman [11], and Cooper and Steinberg [13].

Application

The example cited here arises from the authors' study [14] of a control law presented in 1973 by Kim and Grider [15]. This law was designed to minimize both the miss distance and the body attitude angle at impact for an air to ground missile pursuing a ground target moving at a constant speed. The state equation used was:

$$\begin{bmatrix} \dot{Y}_d \\ \ddot{Y}_d \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{Y}_d \\ -\frac{K_1}{W_1} u \cos\theta \\ Ku \end{bmatrix}$$

where

Y_d = the distance between the missile and the target, projected on the ground,

θ = the body attitude angle of the missile (measured from the vertical)

K, K_1, W_1 = constants

and u is the controller. The controller u was given as

$$u(t) = [C_y,\ C_{\dot{y}},\ C_\theta] \begin{bmatrix} Y_d \\ \dot{Y}_d \\ \theta \end{bmatrix}$$

with

$$C_y = (-\alpha_2 g\Omega - \alpha_1 gK^2\Omega^2/2)/\Delta$$

$$C_{\dot{y}} = (-\alpha_2 g\Omega^2 - \alpha_1 gK^2\Omega^3/2)/\Delta$$

$$C_\theta = (-\alpha_1\alpha_2 K + \alpha_1 Kg^2\Omega^3/6)/\Delta$$

$$\Delta = \alpha_2^2 + \alpha_1\alpha_2 K^2\Omega + \alpha_2 g^2\Omega^3/3 + \alpha_1 g^2K^2\Omega^4/12$$

where

$g \equiv -K_1 \cos\theta_f / W_1$ V = the velocity of missile

$\Omega \equiv t_f - t = -H/(V^* \cos\theta)$ t = time

H = the vertical height of missile t_f = time of impact .

The controller was designed to minimize the cost functional:

$$F = Y_d^2(t_f) + \alpha_1 \theta^2(t_f) + \alpha_2 \int_{t_o}^{t_f} u^2(t)dt \ .$$

Kim and Grider reported a miss distance of 0.045 feet with an attitude angle at impact of 0.115 degrees. They used the following initial values to obtain these results:

H = 10000 feet $K_1 = 1$

Y_d = 10000 feet K = 0.0005

θ = 45 degrees $\alpha_1 = 3823$

V = 2000 ft./second $\alpha_2 = 6.94 \times 10^{-5}$.

W_1 = 5 rad./second

The authors wished to produce similar results for miss distance and attitude angle at impact while varying H, Y_d, θ, and V. The control law was also modified so that the attitude angle at impact θ_f could be specified. The constants W_1, K_1, and K were to remain unchanged. The parameters α_1 and α_2 had to be estimated to achieve the desired result.

To obtain acceptable values for α_1 and α_2, the following nonlinear programming problem was formulated,

$$\min J(\bar{\alpha}) = Y_d^2(\bar{\alpha}) + (\theta(\bar{\alpha}) - \theta_f)^2 \qquad (2 \text{ bis})$$

where values of $Y_d(\bar{\alpha})$ and $\theta(\bar{\alpha})$ would be determined by running the simulation with $\bar{\alpha} = [\alpha_1, \alpha_2]^T$. The Hooke and Jeeves algorithm was selected as the method to be

used in solving the problem, since any algorithm relying on derivative information for $J(\bar{\alpha})$ had to be excluded. The Table gives the results obtained for the initial conditions:

$H = 1000$ feet $\qquad V = 1095$ ft./sec.

$Y_d = 5000$ feet $\qquad \theta_f = 45^\circ$.

$\theta = 85^\circ$

The values of W_1, K_1, and K were the same as those used by Kim and Grider.

	Initial Values	Final Values
α_1	3823.598	5525.508
α_2	0.340E-4	0.190E-6
$J(\bar{\alpha})$	1520.600	10.970
miss distance (feet)	36.950	3.301
attitude impact angle (degrees)	57.462	45.270

Table: Computational Results

The column labeled Initial Values gives the initial choices of α_1 and α_2 as well as the resulting objective function value, the miss distance, and the attitude impact angle. The Final Values column shows the improved values of α_1 and α_2 obtained after repeated applications of the Hooke and Jeeves algorithm. Note that the final attitude impact angle was 45.270°, just 0.27° over the desired impact angle of $\theta_f = 45^\circ$.

Before leaving this example, a computational note is in order. The Final Values reported in the Table were achieved by executing the Hooke and Jeeves algorithm, evaluating the results, and reexecuting the algorithm using the best results from the previous run. This series of steps was repeated several times before the Final Values were obtained. This is not unusual behavior for any algorithm to exhibit since the choice of search directions and step lengths used by an algorithm may cause criteria (4b) to be satisfied before a minimum

value of the objective function is obtained. Restarting the algorithm will usually be successful in the further refinement of the parameters being estimated.

Conclusions

Nonlinear programming algorithms are a valuable tool in the determination of many control law parameters. This paper has presented a review of several direct search algorithms and an example of how these algorithms can be applied to obtain parameter estimates.

As can be seen in the control problem example, the problem of finding optimal values for the control parameters α_1 and α_2 is replaced with a mathematical programming problem; however, this new problem still has hidden in it two unknown search parameters t_1 and t_2. The advantage of this formulation is that there are several techniques available for finding t_1 and t_2 as discussed in the paragraph just above the example.

In applying the algorithms, the approach is essentially the same. Specify the parameters to be estimated and the objective function to be optimized. The nonlinear programming algorithm then uses values of the objective function at various points to produce improved parameter estimates. The process is repeated until no additional improvement can be made in parameter values or until the objective function reaches a specified value.

REFERENCES

1. Bass, R. "A Rank Two Algorithm for Unconstrained Minimization," Mathematics of Computation, Vol. 26, 1972, pp. 129-143.

2. Fletcher, R. "A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 13, 1970, pp. 317-322.

3. Levenberg, K. "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quarterly of Applied Mathematics, Vol. 8, 1944, pp. 212-221.

4. Marquardt, Donald W. "An Algorithm for Least-Squares Estimation of Non-Linear Parameters," Journal of the Society of Industrial Applied Mathematics, Vol. 2, 1963, pp. 164-168.

5. Himmelblau, D.M. Applied Nonlinear Programming. New York: McGraw-Hill, 1972.

6. Hooke, R. and Jeeves, T.A. "Direct Search Solution of Numerical and Statistical Problems," Journal of the Association for Computing Machinery, Vol. 8, 1961, pp. 212-229.

7. Rosenbrock, H.H. "An Automatic Method for Finding the Greatest or Least Value of a Function," The Computer Journal, Vol. 3, 1960, pp. 175-184.

8. Rice, J.R. "Experiments of Gram-Schmidt Orthogonalization," Mathematics of Computation, Vol. 20, 1966, pp. 325-328.

9. Box, M.J., D. Davies and W.H. Swann. Nonlinear Optimization Techniques: I. C. I. Monograph No. 5. Edinburgh, Scotland: Oliver and Boyd, Ltd., 1969.

10. Powell, M.J.D. "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives." Computer Journal, Vol. 7, pp. 155-162, 1964.

11. Lessman, R.E. Mixed Nonderivative Algorithms for Unconstrained Optimization. Rolla: University of Missouri-Rolla, 1975.

12. Zangwill, W.I. "Minimizing a Function Without Calculating Derivatives." Computer Journal, Vol. 10, pp. 293-296, 1967.

13. Cooper, L. and Steinberg, D. Introduction to Methods of Optimization. Philadelphia: W.B. Saunders, 1970.

14. Pastrick, H.L., York, R.J., and St. Clair, D.C. "On the Realization of an Optimal Control Law for a Terminally Guided Missile," Proceedings of the Tenth Annual Southeastern Symposium on System Theory, March, 1978, pp. IV-A-38 - IV-A-48.

15. Kim, M. and Grider, K.V. "Terminal Guidance for Impact Attitude Angle Constrained Flight Trajectories," IEEE Transaction on Aerospace and Electronic Systems, Vol. AES-9, 1973, pp. 852-859.

Appendix D: Hooke-Jeeves Algorithm Listing

```
      REAL*8 CQUANT
      COMMON NBASE,X,N,LA,K,KK,NPRNT,NTRY
      COMMON/CONSTR/CQUANT
      EXTERNAL COMPUT
      DIMENSION XO(10),X(10),XMAX(10),XMIN(10)
      NTRY=0
      NBASE=0
      READ(5,10000)TOL,ALPH,BETA,DEL,LIM,LT,LSN,N,NPRNT
10000 FORMAT(4E12.0,I10,4I4)
      READ(5,10001)(XO(I),XMAX(I),XMIN(I),I=1,N)
10001 FORMAT(3E15.0)
C
C INPUT THETAO FOR KGCTRL FUNCTION
      READ(5,10001) CQUANT
C
5     DO 10 I=1,N
10    X(I)=XO(I)
      LA=0
15    CALL PATTRN(COMPUT,  XMAX,XMIN,          LT,LIM,TOL,LSN,DEL,
     1  ALPH,BETA)
      READ(5,10002,END=999)L,ALPH,BETA,DEL,LA
10002 FORMAT(4E12.0,I1)
      IF (LA.EQ.9) GO TO 15
      IF (LA.GT.0) GO TO 5
  999 STOP
      END
      SUBROUTINE PATTRN(COMPUT,  XMAX,XMIN,          LT,LIM,TOL,LSN,
     1DEL,ALPH,BETA)
      COMMON NBASE,X,N,LA,K,KK,NPRNT,NTRY
      DIMENSION X(10),XMAX(10),XMIN(10)
      DIMENSION C(100),D(100)
      LOGICAL TYPE
C X IS ARRAY OF INDEPENDENT VARIABLES
C C IS ARRAY OF INDEPENDENT VARIABLES AT LAST BASE POINT
C XMAX IS ARRAY OF MAXIMUM LIMITS ON X
C XMIN IS ARRAY OF MINIMUM LIMITS ON X
C D IS ARRAY OF STEP SIZES FOR X
C LA IS PRIMARY BRANCHING CONTROL
C      = 1 FOR INITIALIZATION
C      = 2 FOR INITIAL MOVE OF X(K) IN EXPLORATORY SEARCH
C      = 3 FOR REVERSE MOVE OF X(K) IN EXPLORATORY SEARCH
C      = 4 FOR PATTERN MOVE
C      = 5 FOR INITIAL MOVE OF X(K) AFTER PATTERN MOVE
C      = 6 FOR REVERSE MOVE OF X(K) AFTER PATTERN MOVE
C      = 7 FOR BASE POINT
C      = 8 WHEN SEARCH IS FINISHED
C K IS INDEX OF X
C KK IS COUNTER FOR NO. X,S TESTED SINCE LAST BASE POINT
C N IS NO. OF INDEPENDENT VARIABLES
C DEL IS INITIAL STEP SIZE CONTROL = 0.01 FOR 0 INPUT
C LT IS CONTROL ON MAX KK BEFORE BASE POINT
C      = N-1 FOR 0 INPUT
C      = N FOR INPUT GREATER THAN N
C LIM IS MAX NO OF MOVES = 1.0E5 FOR 0 INPUT
C TOL IS BASE POINT TEST TOLERANCE AND MIN LIMIT ON STEP SIZE
C      = 1.0E-5 FOR 0 INPUT
```

```
C LSN IS CONTROL ON FUNCTION EVALUATION AT BASE POINT
C      EVALUATION IS DONE IF LSN.NE.0
C ALPH IS RATIO TO INCREASE D(K)
C      = 2.5 FOR 0 INPUT
C BETA IS RATIO TO DECREASE D(K)
C      = 1/3 FOR 0 INPUT
       TYPE=.FALSE.
       IF(LA.EQ.9)GO TO 800
       LA=1
100    NCT=0
       DO 14 L=1,N
       IF (XMAX(L).GE.XMIN(L)) GO TO 10
       WRITE(6,2000)L
2000   FORMAT(1H0,68H0   INPUT DATA ERROR.  LIMITS REVERSED FOR INDEPENDE
      1NT VARIABLE NO. ,I4)
       LA=10
       GO TO 500
10     IF (XMAX(L).EQ.XMIN(L)) NCT=NCT+1
       IF (X(L)-XMAX(L)) 12,14,11
11     X(L)=XMAX(L)
       GO TO 14
12     IF (XMIN(L)-X(L)) 14,14,13
13     X(L)=XMIN(L)
14     CONTINUE
       IF (NCT.LT.N) GO TO 15
       LA=1
       CALL COMPUT(SN)
       WRITE(6,2010)
2010   FORMAT(1H0,47H0  EACH PAIR OF UPPER AND LOWER LIMITS IS EQUAL    /
      1  26H      NO SEARCH IS POSSIBLE    )
       LA=10
       GO TO 500
15     IF (LT) 17,17,16
16     IF (LT.GT.N) LT=N
       GO TO 20
17     LT=N-1
       IF (LT.LT.1) LT=1
20     IF (LIM.LE.0) LIM=1E5
       IF (TOL.LE.0) TOL=1.0E-5
       IF (DEL.LE.0) DEL=0.01
       IF (ALPH.LE.0) ALPH=2.5
       IF (BETA.LE.0) BETA=1.0/3.0
800    DO 810 L=1,N
       D(L)=(XMAX(L)-XMIN(L))*DEL
810    C(L)=X(L)
       WRITE(6,3000) (D(L), L=1,N)
 3000  FORMAT('0$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$'/
      #' INITIAL STEP SIZES'/' ',8E15.7/ ' $$$$$$$$$$$$$$$$$$$$$$$')
       NOFAIL=0
       NPF=0
       NCT=1
       K=1
       KK=1
       M1=1
       M2=1
       CALL COMPUT(SN)
       SP=SN
       SC=SN
       LA=1
       GO TO 500
```

```
110    TYPE=.FALSE.
120    LA=2
       GO TO 150
130    IF (.NOT.TYPE) GO TO 120
140    LA=5
150    IF (D(K).EQ.0) GO TO 340
       X(K)=X(K)+D(K)
160    IF (X(K).GT.XMAX(K).OR.X(K).LT.XMIN(K))
      1  GO TO (500,310,330,500,310,330,500,500),LA
       GO TO 850
200    IF (SN-SP) 201,310,310
201    IF (TYPE) D(K)=D(K)*ALPH
210    SP=SN
       NPF=0
       M1=1
       M2=1
220    K=K+1
       IF (K.GT.N) K=1
222    IF (LT.LE.1) GO TO 230
223    IF (KK.LT.LT) GO TO 224
       GO TO 230
224    KK=KK+1
       GO TO 130
230    IF (SP+TOL*ABS(SC).GE.SC) GO TO 234
231    IF ( TYPE.AND.(NPF.LT.5) .OR..NOT.TYPE) GOTO232
       GOTO234
232    LA=7
       NBASE=NBASE+1
       M1=1
       IF (LSN) 850,520,850
234    KK=KK+1
       IF (KK-N) 130,130,235
235    NOFAIL=NOFAIL+1
       IF (NOFAIL-15)280,500,500
280    IF (.NOT.TYPE) GOTO281
282    NPF=0
       DO 283 I=1,N
283    X(I)=C(I)
       GO TO 284
281    IF (M1.GT.N) GO TO 300
284    KK=1
       M1=1
       M2=1
       SP=SC
       GO TO!110
  300  WRITE(6,2020)SC
       GO TO 500
310    IF (TYPE) GOTO312
311    LA=3
       GO TO 313
312    LA=6
313    X(K)=X(K)-D(K)-D(K)
       GO TO 160
320    IF (SN.GE.SP) GOTO330
321    D(K)=-D(K)
       GO TO 210
330    X(K)=X(K)+D(K)
       DX=TOL*ABS(X(K))
       IF (DX.LT.1.0E-30) DX=1.0E-30
332    D(K)=D(K)*BETA
```

```
      IF (ABS(D(K)).GE.DX) GOTO340
333   D(K)=SIGN(DX,D(K))
      M1=M1+1
340   IF (.NOT.TYPE) GO TO 220
342   M2=M2+1
      IF (M2.LE.N) GO TO 220
343   M2=1
      NPF=NPF+1
      GO TO 220
C      END MINIMUM STEP SIZE TESTS AND PATTERN FAILURE COUNT
500   CONTINUE
2020  FORMAT(///5H  SN=,E15.7,5H  SP=,E15.7,5H  SC=,E15.7,6H  DEL=,
     1 E15.7,/6H  TOL=,E15.7,8H  ALPHA=,E15.7,7H  BETA=,E15.7/
     18X,8H  TYPE  ,L1,3X,3H N=,I3,3X,3H K=,I3,3X,4H KK=,I4,3X,4H LA=,
     1I3,/ 8X,4H LT=,I3,3X,5H LSN=,I3,3X,4H M1=,I3,3X,4H M2=,I3,3X,
     15H NPF=,I3,/ 8X,5H NCT=,I6,3X,5H LIM=, I7,3X,7H NBASE=,I4,3X,
     18H NOFAIL=,I3,//  3H NO,9X,1HX,18X,1HC,18X,1HD,16X,4HXMAX,15X,
     14HXMAX,/)
2030  FORMAT (I3,5E19.8)
      WRITE(6,2020)SN,SP,SC,DEL,TOL,ALPH,BETA,TYPE,N,K,KK,LA,LT,LSN,M1,M
     12, NPF,NCT,LIM,NBASE,NOFAIL
      DO 501 I=1,N
501   WRITE(6,2030) I,X(I),C(I),D(I),XMAX(I),XMIN(I)
C      OUTPUT BLOCK FOR LA ERROR ENTRY AND LIM FINISH
      IF (LA.EQ.1) GO TO 110
505   IF (LA.GE.8) GOTO900
506   LA=8
      DO 507 I=1,N
507   X(I)=C(I)
      GO TO 850
510   IF (LSN) 511,520,511
511   SP=SN
520   SC=SP
      LA=4
      KK=1
      DO 526 L=1,N
      P = C(L)
      C(L) = X(L)
      IF ((2*X(L)-P).LE.XMAX(L)) GOTO523
522   X(L)=XMAX(L)
      GO TO  526
523   IF((2*X(L)-P).GE.XMIN(L)) GOTO525
524   X(L)=XMIN(L)
      GO TO 526
525   X(L)=2*X(L)-P
526   CONTINUE
527   NOFAIL=0
      GO TO 850
530   SP=SN
      TYPE=.TRUE.
      GO TO 140
850   NCT=NCT+1
      IF(NCT-LIM) 852,851,851
851   IF(LA-8)  500,852,500
852   CALL COMPUT(SN)
C
C  IF MOVE SUCCESSFUL, CHECK HARD CONSTRAINTS--ELSE CONTINUE.
C
      GO TO 1000
      IF (SP .LT. SN) GO TO 1000
```

```
      IF (ABS(ABS(CQUANT)-45.) .LE.  5.0) GO TO 1000
      WRITE(6,1002) CQUANT,LA,NCT,(X(I4), I4=1,N)
 1002 FORMAT('0', 50('#'),/' HARD CONSTRAINT VIOLATED: CQUANT =',
     %G9.4, ' LA = ',I2, 'NCT = ',I5,/' X= ',12G9.4)
      WRITE(6,1003) (D(I4), I4=1,N)
 1003 FORMAT(' ARRAY OF STEPLENGTHS:'/' ', 10G11.4)
C  IF CONSTRAINT VIOLATED, FAKE INCREASE IN FCTN. VALUE AND LET ORIGINAL
C  CODE SHORTEN STEP LENGTH.
      SN = SP + 1.0
C
 1000 GO TO (100,200,320,530,200,320,510,900,800),LA
900   RETURN
      END
      SUBROUTINE COMPUT(Z)
      REAL KGCTRL
      COMMON NBASE,X,N,LA,K,KK,NPRNT,NTRY
      DIMENSION X(10)
C     YY= 100.0*(X(2)-X(1)**2)**2+(1-X(1))**2
C  CALL KIM-GRIDER CONTROL LAW
      YY=ABS(KGCTRL(X(1),X(2),NBASE))
      NTRY=NTRY+1
      IF (LA.EQ.7) NBASE=NBASE+1
      IF (LA.EQ.1 .OR. LA.GT.6 .OR. NPRNT.NE.0)
     1WRITE(6,5)YY,LA,K,KK,NTRY,NBASE,(I,X(I),I=1,N)
5     FORMAT( 4H0 SN,E15.7,3H LA,I2,2H K,I2,3H KK,I2,5H NTRY,I5,
     1 6H NBASE,I5/(4(4H  X(,I1,1H),E15.7)))
      Z=YY
      RETURN
      END
..INC KGCTRL DBLFCTN
//GO.SYSIN  DD  *
 0.00001       1.5         0.050          0.01000         0500   3   2   2
6199.992        6200.          0.
.00002351592    .1             0.
87.5
/*
```

Appendix E: Derivation of the Three-State Controller

Problem: Derive a control law of the form

$$u = c_1(t)Y_d + c_2(t)\dot{Y}_d + c_3(t)\theta \qquad \text{(E-1)}$$

to minimize the cost functional

$$J = Y_d^2(t_f) + \gamma\theta^2(t_f) + \beta\int_{t_o}^{t_f} u^2(t)dt \qquad \text{(E-2)}$$

where the state variables Y_d, $\dot{Y}_d$, θ are subject to the dynamics

$$\dot{Y}_d = \dot{Y}_d \qquad \text{(E-3)}$$

$$\ddot{Y}_d = -(K_1/W_1)ub \quad , \quad b = \cos\theta(t_f) \qquad \text{(E-4)}$$

$$\dot{\theta} = K_a \cdot u \quad . \qquad \text{(E-5)}$$

<u>Solution</u>: Using the technique of Lagrangian multipliers, define the Hamiltonian to be

$$H \equiv \beta \cdot u^2 + \lambda_1\dot{Y}_d - \lambda_2 K_1 bu/W_1 + \lambda_3 K_a u \quad . \qquad \text{(E-6)}$$

Following the method of solution as outlined in [1], in order for u to be the optimal controller,

$$\partial H/\partial u = 0 = 2\beta u - \lambda_2 K_1 b/W_1 + \lambda_2 K_a \qquad \text{(E-7)}$$

and so

$$u = [\lambda_2 K_1 b/W_1 - \lambda_3 K_a]/(2\beta) \quad . \qquad \text{(E-8)}$$

The controller u then is determined once the Lagrangian multipliers are known.

<u>Determination of</u> λ_1, λ_2, λ_3

The condition that they must satisfy is

$$\partial H/\partial X_i = -\dot{\lambda}_i \quad , \quad i = 1,2,3 \qquad \text{(E-9)}$$

where

$$X_i = i^{th} \text{ state} \quad , \quad i = 1,2,3 \quad .$$

Thus,

$$\partial H/\partial X_1 = \partial H/\partial Y_d = 0 = -\dot{\lambda}_1 \tag{E-10}$$

$$\partial H/\partial X_2 = \partial H/\partial \dot{Y}_d = \lambda_1 - \dot{\lambda}_2 \tag{E-11}$$

$$\partial H/\partial X_3 = \partial H/\partial \theta = 0 = -\dot{\lambda}_3 \ . \tag{E-12}$$

Boundary conditions for λ_1, λ_2, λ_3 are given by:

$$\lambda_i^T(t_f) = \partial u/\partial X_i(t_f) \ , \ i = 1,2,3 \ . \tag{E-13}$$

Thus,

$$\lambda_1(t_f) = 2\cdot Y_d(t_f) \tag{E-14}$$

$$\lambda_2(t_f) = 0 \tag{E-15}$$

$$\lambda_3(t_f) = 2\cdot \gamma\theta(t_f) \ . \tag{E-16}$$

Solving the differential equation (E-10) and using condition (E-14), we get

$$\lambda_1(t) = 2\cdot Y_d(t_f) \ . \tag{E-17}$$

Equation (E-11) combined with Equation (E-17) yields

$$\lambda_2(t) = -2\cdot Y_d(t_f)t + c \ . \tag{E-18}$$

Boundary Condition (E-15) implies

$$\lambda_2(t) = -2\cdot Y_d(t_f)t + 2\cdot Y_d(t_f)t_f \ . \tag{E-19}$$

For λ_3, Equation (E-12) combined with condition (E-16) gives

$$\lambda_3(t) = 2\cdot \gamma\theta(t_f) \ . \tag{E-20}$$

Hence, the control u given by Equation (E-8) becomes

$$u(t) = [g\cdot Y_d(t_f)(t-t_f) - K_a\cdot \gamma\theta(t_f)]/\beta \tag{E-21}$$

where

$$g \equiv -K_1 b/W_1 \ . \tag{E-22}$$

The controller can now be determined if $Y_d(t_f)$ and $\theta(t_f)$ are known in closed form.

Determination of $Y_d(t_f)$ and $\theta(t_f)$

Once the form of the controller has been determined as in Equation (E-21), we can return to the original system of differential equations (E-3) - (E-5) and solve for $Y_d(t)$, $\theta(t)$. Equation (E-4) becomes

$$\ddot{Y}_d = gu = g[gY_d(t_f)(t-t_f) - K_a \cdot \gamma\theta(t_f)]/\beta \ . \tag{E-23}$$

Integrating,

$$\dot{Y}_d = g[gY_d(t_f)(t-t_f)^2/2 - K_a \cdot \gamma\theta(t_f)(t-t_f) + d_1]/\beta \tag{E-24}$$

and again,

$$Y_d(t) = g[gY_d(t_f)(t-t_f)^3/6-K_a \cdot \gamma\theta(t_f)(t-t_f)^2/2 + d_1(t-t_f) + d_2]/\beta \ . \tag{E-25}$$

Also, for $\theta(t)$,

$$\dot{\theta} = K_a[g \cdot Y_d(t_f)(t-t_f) - K_a \cdot \gamma\theta(t_f)]/\beta$$

so

$$\theta(t) = K_a[g \cdot Y_d(t_f)(t-t_f)^2/2 - K_a \gamma\theta(t_f)(t-t_f) + d_3]/\beta \ . \tag{E-26}$$

Now, letting $t = t_o$ we determine d_1, d_2, d_3 from Equations (E-24), (E-25), (E-26). Substituting and simplifying yields:

$$d_1 = -(t_o-t_f)^2 g \cdot Y_d(t_f)/2 + (t_o-t_f)Ka \cdot \gamma\theta(t_f) + \beta\dot{Y}_d(t_f)/g \tag{E-27}$$

$$\begin{aligned} d_2 = {} & (t_o-t_f)^3 g \cdot Y_d(t_f)/3 + (t_o-t_f)^2(-K_a \cdot \gamma\theta(t_f))/2 \\ & + (t_o-t_f)(-\beta\dot{Y}_d(t_f)/g) + \beta Y_d(t_o)/g \end{aligned} \tag{E-28}$$

$$d_3 = (t_o-t_f)^2(-g \cdot Y_d(t_f))/2 + (t_o-t_f)K_a \cdot \gamma\theta(t_f) + \beta\theta(t_o)/K_a \ . \tag{E-29}$$

Using d_1, d_2, d_3 and now letting $t=t_f$, we can write down two equations--two unknowns: $Y_d(t_f)$ and $\theta(t_f)$.

$$Y_d(t_f) = g \cdot d_2/\beta$$

$$\begin{aligned} Y_d(t_f) = {} & (t_o-t_f)^3 Y_d(t_f) g^2/(3\beta) + (t_o-t_f)^2(-K_a \cdot \gamma\theta(t_f)g)/ \\ & (2\beta) + (t_o-t_f)(-\dot{Y}_d(t_o) + Y_d(t_o) \end{aligned} \tag{E-30}$$

and

$$\theta(t_f) = K_a d_3/\beta$$

$$\theta(t_f) = (t_o - t_f)^2(-gK_a Y_d(t_f))/(2\beta) + (t_o - t_f)K_a^2\gamma\theta(t_f)/\beta + \theta(t_o) \ . \qquad \text{(E-31)}$$

Solving Two Equations - Two Unknowns

The system for $Y_d(t_f)$, $\theta(t_f)$ is (with $\Omega = t_o - t_f$):

$$Y_d(t_f)[1-g^2\Omega^3/(3\beta)] + \theta(t_f)[K_a\gamma g\Omega^2/(2\beta)] = Y_d(t_o) - \Omega\dot{Y}_d(t_o) \qquad \text{(E-32)}$$

and

$$Y_d(t_f)[gK_a\Omega^2/(2\beta)] + \theta(t_f)[1-K_a^2\gamma\Omega/\beta] = \theta(t_o) \ . \qquad \text{(E-33)}$$

Using Crammer's Rule to solve, we calculate the determinant of the coefficient matrix:

$$\det A = (1/\beta^2)[\gamma g^2 K_a^2\Omega^4/12 + \beta g^2\Omega^3/3 + \Omega\gamma\beta K_a^2 + \beta^2] \ . \qquad \text{(E-34)}$$

Defining

$$\Delta \equiv \beta^2 \det A \quad , \qquad \text{(E-35)}$$

we obtain the following:

$$\begin{aligned} Y_d(t_f) = \{&\beta^2 Y_d(t_o) + \Omega[-\beta^2\dot{Y}_d(t_o) - \beta K_a^2\gamma Y_d(t_o)] \\ &+ \Omega^2[2\beta\dot{Y}_d(t_o)K_a^2\gamma - \theta(t_o)K_a\gamma g\beta]/2\}/\Delta \end{aligned} \qquad \text{(E-36)}$$

$$\begin{aligned} \theta(t_f) = \{&\beta^2\theta(t_o) - \beta gK_a Y_d(t_o)\Omega^2/2 + \\ &[3\beta gK_a\dot{Y}_d(t_o) - 2g^2\beta\theta(t_o)]\Omega^3/6\}/\Delta \ . \end{aligned} \qquad \text{(E-37)}$$

Therefore, returning to Equation (E-21),

$$u(t;t_o) = gY_d(t_f)\Omega/\beta - K_a\gamma\theta(t_f)/\beta, \qquad \text{(E-38)}$$

and substituting Equations (E-36), (E-37) and simplifying, yields

$$\begin{aligned} u(t) = &Y_d(t)[-g\beta\Omega - gK_a^2\gamma\Omega^2 + gK_a^2\gamma\Omega^2/2]/\Delta \\ &+ \dot{Y}_d(t)[-g\beta\Omega^2 - gK_a^2\gamma\Omega^3 + K_a^2\gamma g\Omega^3/2]/\Delta \\ &+ \theta(t)[K_a\gamma g^2\Omega^3/2 - K_a\gamma\beta - \gamma K_a g^2\Omega^3/3]/\Delta \ . \end{aligned} \qquad \text{(E-39)}$$

Hence, we conclude that

$$c_1(t) = [-g\beta(t_f-t) - gK_a^2\gamma(t_f-t)^2/2]/\Delta \tag{E-40}$$

$$c_2(t) = [-g\beta(t_f-t)^2 - gK_a^2\gamma(t_f-t)/2]/\Delta \tag{E-41}$$

$$c_3(t) = [K_a\gamma g^2(t_f-t)^3/6 - K_a\gamma\beta]/\Delta \tag{E-42}$$

where

$$\Delta = \gamma g^2 K_a^2 (t_f-t)^4/12 + \beta g^2(t_f-t)^3/3 + \gamma\beta K_a^2(t_f-t) + \beta^2 \tag{E-43}$$

and

$$g = -K_1 b/W_1 \ .$$

DISTRIBUTION

ARMY	No. of Copies
Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12
Commander ASD ATTN: DRCPH-HFF, Col. Dobbs Wright-Patterson Air Force Base, Ohio 45433	1
Commander Picatinny Arsenal ATTN: DRCPM-CAWS-PA, Col. R. Phillips Dover, New Jersey 07801	1

LOCAL

DRDMI-T	Dr. Kobler	1
DRDMI-TG	Mr. Huff	1
DRDMI-TGN	Mr. Gambill	1
	Dr. Pastrick	2
DRCPM-HF	Col. Fiest	1
DRCPM-HFE	Mr. Comer	1